Clustering-based collocation for uncertainty propagation with multivariate correlated inputs

A.W. Eggels[a,*], D.T. Crommelin[a,b], J.A.S. Witteveen[a]

[a] *Centrum Wiskunde & Informatica, Amsterdam, the Netherlands*
[b] *Korteweg - de Vries Institute for Mathematics, University of Amsterdam, the Netherlands*

Abstract

In this article, we propose the use of partitioning and clustering methods as an alternative to Gaussian quadrature for stochastic collocation (SC). The key idea is to use cluster centers as the nodes for collocation. In this way, we can extend the use of collocation methods to uncertainty propagation with multivariate, correlated input. The approach is particularly useful in situations where the probability distribution of the input is unknown, and only a sample from the input distribution is available. We examine several clustering methods and assess their suitability for stochastic collocation numerically using the Genz test functions as benchmark. The proposed methods work well, most notably for the challenging case of nonlinearly correlated inputs in higher dimensions. Tests with input dimension up to 16 are included.

Furthermore, the clustering-based collocation methods are compared to regular SC with tensor grids of Gaussian quadrature nodes. For 2-dimensional uncorrelated inputs, regular SC performs better, as should be expected, however the clustering-based methods also give only small relative errors. For correlated 2-dimensional inputs, clustering-based collocation outperforms a simple adapted version of regular SC, where the weights are adjusted to account for input correlation.

Keywords: uncertainty quantification, stochastic collocation, correlated input distributions, partitioning, clustering, k-means, Principal Component Analysis.

1. Introduction

A core topic in the field of uncertainty quantification (UQ) is the question how uncertainties in model inputs are propagated to uncertainties in model outputs. How can one characterize the distribution of model outputs, given the distribution of the model inputs (or a sample thereof)? Questions such as

*Corresponding author
Email addresses: a.w.eggels@cwi.nl (A.W. Eggels), Daan.Crommelin@cwi.nl (D.T. Crommelin)

these are encountered in many fields of science and engineering [1, 2, 3, 4, 5], and have given rise to modern UQ methods such as stochastic collocation (SC), polynomial chaos expansion (PCE), generalized polynomial chaos (gPC) and stochastic Galerkin methods [6, 7, 8, 9, 10].

A still outstanding challenge is how to characterize model output distributions efficiently in case of multivariate, correlated input distributions.

In the previously mentioned methods independence between the inputs is assumed, e.g., for the construction of the Lagrange polynomials in SC, or for the construction of the orthogonal polynomials in gPC. When independence between the input components holds, the multivariate problem can easily be factored into multiple 1-dimensional problems, whose solutions can be combined by tensor products to a solution for the multidimensional problem. When the inputs are not independent, such factorization can become extremely complicated, making it unfeasible in practice for many cases. If possible at all, it generally involves nontrivial transformations that require detailed prior knowledge of the joint distribution (e.g. Rosenblatt transformation). In [11], factorization is circumvented and instead the problem is tackled by using the Gram-Schmidt (GS) orthogonalization procedure to get an orthogonal basis of polynomials, in which the orthogonality is with respect to the distribution of the inputs. However, this procedure gives non-unique results that depend on the implementation.

Clearly, the case of correlated inputs can be handled, in principle, with straightforward Monte Carlo sampling (MCS). In practice, considerations of efficiency and computational cost often preclude MCS. UQ methods are designed to be much more efficient than MCS, however the efficiency gain can be strongly reduced (or even vanish) if the dimension of the input vector increases. Methods that suffer from curse of dimension become unfeasible for high-dimensional problems.

In this paper we propose a novel approach for efficient UQ with multivariate, correlated inputs. This approach is based on stochastic collocation, however it employs collocation nodes that are obtained from data clustering rather than from constructing a standard (e.g. Gaussian) quadrature or cubature rule. By using techniques from data clustering, we can construct sets of nodes that give a good representation of the input data distribution, well capable of capturing correlations and nonlinear structures in the input distributions. Clustering also leads naturally to weights associated with the nodes.

The approach we propose is able to handle correlated inputs and, as we demonstrate, remains efficient for higher dimensions of the inputs, notably in case of strong correlations. It is non-intrusive, as it is in essence a stochastic collocation method. Thus, existing deterministic solvers to compute model outputs from inputs can be re-used. Furthermore, the approach employs data clustering, starting from a sample dataset of inputs. The underlying input distribution can be unknown, and there is no fitting of the distribution involved. This makes the approach very suitable for situations where the exact input distribution is unknown and only a sample of it is available.

The outline of this paper is the following: in Section 2, we briefly summarize stochastic collocation and multivariate inputs. We discuss the challenges of

dealing with correlated inputs, and we introduce the concept of clustering-based collocation. In Section 3, we describe three different clustering techniques. In Section 4, we present results of numerical experiments in which we test our clustering-based collocation method, using the clustering techniques described in Section 3. In Section 5, these methods are compared against standard SC methods for uncorrelated inputs, in which the data set is constructed through a Gaussian copula. The conclusion follows in Section 6.

2. Stochastic collocation and its extension

Consider a function $u(x) : \Omega \mapsto \mathbb{R}\,, \quad \Omega \subseteq \mathbb{R}^p$, that maps a vector of input variables to a scalar output. A typical question for uncertainty quantification is how uncertainty in the input(s) x propagates to uncertainty in the output $u(x)$. More specifically, let us assume x is a realization of a random variable X with probability density function $f(x)$. We would like to characterize the probability distribution of $u(x)$, in particular we would like to compute moments of $u(x)$:

$$\mathbb{E}[u^q] = \int_\Omega (u(x))^q f(x) dx\,. \tag{1}$$

In what follows, we focus on the first moment:

$$\mu := \mathbb{E}\, u = \int_\Omega u(x) f(x) dx\,. \tag{2}$$

We note that higher moments can be treated in the same way, as these are effectively averages of different output functions, i.e. $\mathbb{E}[u^q] = \mathbb{E}v$ with $v(x) := (u(x))^q$. In both cases, the expectation is with respect to the distribution of X.

In stochastic collocation (SC), the integral in (2) is approximated using a quadrature or cubature rule. As is well-known, a high degree of exactness of the integration can be achieved for polynomial integrands with Gaussian quadrature rules. Suppose $u(x)$ is approximated by means of Lagrange interpolation, i.e.

$$u(x) \approx \hat{u}(x) := \sum_{i=1}^{k} u(x_i) L_i(x)\,, \tag{3}$$

in which the x_i are the nodes and the $L_i(x)$ are the Lagrange interpolation polynomials satisfying $L_i(x_j) = \delta_{ij}$. Then μ can be approximated as

$$\mu \approx \hat{\mu} := \mathbb{E}\, \hat{u} = \sum_{i=1}^{k} u(x_i) f_i \tag{4}$$

with weights

$$f_i := \int_\Omega L_i(x) f(x) dx \tag{5}$$

The integration is exact, i.e. $\mu - \hat{\mu} = 0$, if $u(x)$ is a polynomial of degree $2k-1$ (or less) and the nodes x_i are those of a Gaussian quadrature rule with $f(x)$ as weight function.

2.1. Multivariate inputs

For multivariate inputs ($p > 1$), SC based on Gaussian quadrature can be constructed using tensor products if the input variables are mutually uncorrelated (independent). In this case, we can write $f(x)$ as a product of 1-dimensional probability density functions:

$$f(x) = f(x_1, \ldots, x_p) = f_1(x_1) f_2(x_2) \cdots f_p(x_p)\,, \tag{6}$$

and we can decompose Ω as

$$\Omega = \Omega_1 \otimes \Omega_2 \otimes \cdots \otimes \Omega_p \tag{7}$$

such that the first moment can be approximated as

$$\hat{\mu} := \mathbb{E}\,\hat{u} = \sum_{i=1}^{k} u(x_{i,1}, \ldots, x_{i,p}) f_i \tag{8}$$

with weights

$$f_i := \int_{\Omega} L_i(x) f(x) dx = \int_{\Omega_p} \cdots \int_{\Omega_1} L_i(x_1, \ldots, x_p) f_1(x_1) \cdots f_p(x_p) dx_1 \cdots dx_p. \tag{9}$$

Because of the decomposition of the domain Ω, and the availability of nodes x_i for each input variable, we can construct multidimensional nodes by a tensor product as well:

$$\{(x_{i,1}, \ldots, x_{i,p})\} := \{x_{i,1}\} \otimes \cdots \otimes \{x_{i,p}\} \tag{10}$$

and similar for the Lagrange interpolation polynomials:

$$L_i(x_{j,1}, \ldots, x_{j,p}) = L_{i,1}(x_{j,1}) \cdots L_{i,p}(x_{j,p}) \tag{11}$$

From this decomposition, we can simplify the computation of the weights f_i

$$\begin{aligned} f_i &= \int_{\Omega_p} \cdots \int_{\Omega_1} L_{i,1}(x_1) \cdots L_{i,p}(x_p) f_1(x_1) \cdots f_p(x_p) dx_1 \cdots dx_p \\ &= \int_{\Omega_1} L_{i,1}(x_1) f_1(x_1) dx_1 \cdots \int_{\Omega_p} L_{i,p}(x_p) f_p(x_p) dx_p \qquad (12) \\ &= f_{i,1} \cdots f_{i,p}. \qquad (13) \end{aligned}$$

to the product of the 1-dimensional weights.

The degree of exactness of this cubature rule is $2k-1$ in each dimension if k nodes are used for each of the input variables. This means that all monomials $x_1^{a_1} x_2^{a_2} \cdots x_p^{a_p}$ are integrated exactly if $0 \leq a_j \leq 2k-1$ for $j = 1, \ldots, p$. Thus, a Gaussian cubature rule in p dimensions is obtained by forming a tensor grid of nodes from 1-dimensional Gaussian quadrature rules, thanks to the mutual independence of the elements of x. It implies for example that when $k = 2$,

$p = 2$, then $x_1^4 x_2^1$ is not integrated exactly, while $x_1^3 x_2^3$ is, despite the lower total degree of the former. Therefore, the degree of exactness in higher dimensions is defined as the minimum of the 1-dimensional degrees of exactness (which is 3 in this example).

An improvement with respect to the curse of dimension is given by Smolyak sparse grids [12], [13]. The construction of Smolyak sparse grids will not be explained in detail here, but an important aspect is that the resulting set of nodes is a union of subsets of full tensor grids. Because the construction contains one-dimensional interpolations with varying numbers of nodes, it works best when the nodes are nested. Therefore, often the Clenshaw-Curtis [14] or Fejér nodes (Clenshaw-Curtis nodes without the boundary nodes) are used, despite the fact that these nodes do not create a Gaussian cubature rule. For an illustration, see Figure 1. In this figure, we show the Gaussian and Clenshaw-Curtis nodes for $p = 2$ and $k = 9$ for both the full and Smolyak sparse construction, for the case of a uniform distribution (i.e., $f(x) = 1$ everywhere on $[0, 1]^2$).

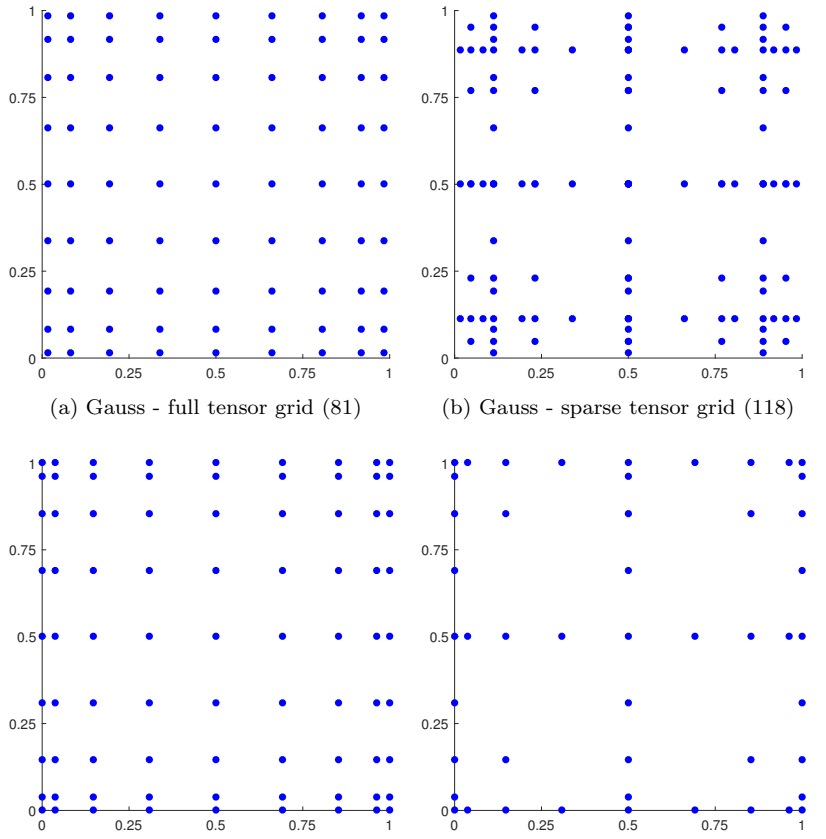
2.2. Gaussian cubature with correlated inputs

As already mentioned, tensor grids are useful for SC in case of uncorrelated inputs. If the input variables are correlated, grids constructed as tensor products of 1-dimensional Gaussian quadrature nodes no longer give rise to a Gaussian cubature rule. In [11], generalization to correlated inputs is approached by constructing sets of polynomials that are orthogonal with respect to general multivariate input distributions, using Gram-Schmidt (GS) orthogonalization. The roots of such a set of polynomials can serve as nodes for a Gaussian cubature rule.

With the approach pursued in [11], the advantageous properties of Gaussian quadrature (in particular, its high degree of exactness) carry over to the multivariate, correlated case. However, one encounters several difficulties with this approach. First of all, for a given input distribution, the set of nodes that is obtained is not unique. Rather, the resulting set depends on the precise ordering of the monomials that enter the GS procedure. For example, with 2-dimensional inputs and cubic monomials, 24 different sets of nodes can be constructed, as demonstrated in [11]. It is not obvious a priori which of these sets is optimal.

A further challenge is the computation of the weights for the cubature rule. It is not straightforward how to construct multivariate Lagrange interpolating functions and evaluate their integrals. The alternative for computing the weights is to solve the moment equations. However, the resulting weights can be negative.

Furthermore, one cannot choose the number of nodes freely: generically, with input dimension p and polynomials of degree m, one obtains m^p nodes. Thus, the number of nodes increases in large steps, for example with $p = 8$ the number of nodes jumps from 1 to 256 to 6561, respectively, if m increases from 1 to 2 to 3. It is unknown how to construct useful (sparse) subsets of nodes from these.



(a) Gauss - full tensor grid (81)

(b) Gauss - sparse tensor grid (118)

(c) Clenshaw-Curtis - full tensor grid (81)

(d) Clenshaw-Curtis - sparse tensor grid (49)

Figure 1: Quadrature points with the number of points in each grid between brackets.

2.3. Clustering-based collocation

To circumvent the difficulties of Gaussian cubature in case of correlated inputs, as summarized in the previous section, we propose an alternative approach to choose collocation nodes. By no longer requiring the collocation nodes to be the nodes of an appropriate Gaussian cubature rule, we loose the benefit of maximal degree of exact integration associated with Gaussian quadrature or cubature. However, we argue below that this benefit of Gaussian cubature offers only limited advantage in practice.

If one has a sample of the inputs available but the underlying input distribution is unknown, the Gaussian cubature rule will be affected by sampling error (via the GS orthogonalization). Alternatively, if the input distribution is estimated from input sample data, the precision of the Gaussian cubature rule is also limited by the finite sample size.

Additionally, the degree of exactness is strongly limited by the number of nodes in higher dimensions. For example, suppose one can afford no more than 256 evaluations of the output function $u(x)$ because of high computational cost, i.e. one can afford a Gaussian cubature rule with 256 nodes. This gives very high degree of integration exactness (degree 511) in one dimension ($p = 1$), but the degree of exactness decreases to 31, 7 and 3, respectively, as the input dimension p increases to 2, 4 and 8. For $p > 8$, the degree of exactness is only 1 in case of 256 nodes, so only linear functions can be integrated exactly.

Instead of constructing a Gaussian cubature rule, we aim to determine a set of nodes that are representative for the sample of input data or for the input distribution, with the locations of the nodes adjusting to the shape of the distribution. Clustering is a suitable method (or rather collection of methods) to achieve this objective.

Clustering is the mathematical problem of grouping a set of objects (e.g., data points) in such a way that objects in one group (or cluster) are more similar to each other than to objects in other clusters. For each of the clusters, a center or medoid can be defined to represent the cluster. This center has the least dissimilarity to other objects in the same cluster, and when the center is chosen to be an object in the cluster, then it is called a medoid.

The basic idea, in the context of this study, is the following. Assume we have a dataset $\{x_1, ..., x_N\}$ available, with $x_i \in \mathbb{R}^p$. We define a collection of K subsets of $\mathbb{R}^p$, denoted Ω_k with $k = 1, ..., K$. A cluster is a subset of the data falling into the same Ω_k. A common way to define cluster centers z_k is as the average of the data in each cluster, i.e.

$$z_k := \frac{\sum_{i=1}^{N} x_i \, \mathbf{1}(x_i \in \Omega_k)}{\sum_{i=1}^{N} \mathbf{1}(x_i \in \Omega_k)} \,, \tag{14}$$

with $\mathbf{1}(\cdot)$ the indicator function. If we define weights w_k as the fraction of all the data falling in the k-th cluster, that is,

$$w_k := \frac{\sum_{i=1}^{N} \mathbf{1}(x_i \in \Omega_k)}{N} \,, \tag{15}$$

the weighted average $\bar{z} := \sum_k w_k \, z_k$ equals the data average $\bar{x} := N^{-1} \sum_i x_i$. Thus, $\bar{z} = \bar{x}$ by construction.

The key idea of what we propose here is to carry out stochastic collocation based on clustering of the input data. More specifically, we propose to use the cluster centers z_k and weights w_k as the nodes and weights of the quadrature rule that underlies stochastic collocation. Thus, the first moment of the input function $u(x)$ over the input data is

$$\mu = \frac{1}{N} \sum_{i=1}^{N} u(x_i) \tag{16}$$

and the approximation using clustering-based collocation is

$$\hat{\mu} := \sum_{k=1}^{K} w_k \, u(z_k) \tag{17}$$

It is easy to show that the approximation is exact ($\hat{\mu} = \mu$) for all linear input functions, due to the fact that $\bar{z} = \bar{x}$, as mentioned above. In other words, the degree of exactness is one: we can consider (17) as a quadrature rule for the integral of $u(x)$ over the empirical measure induced by the dataset $\{x_1, ..., x_N\}$. This quadrature rule is exact if $u(x)$ is linear.

3. Clustering methods

In this section, we summarize three different methods to construct clusters, i.e. three methods to construct a suitable collection of subsets Ω_k. As already mentioned, the methods are based on input given as a dataset in p dimensions with N data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_i \in \mathbb{R}^{1\times p}$ also denoted by a matrix $X \in \mathbb{R}^{N\times p}$. If the input is given as a distribution, we can create a dataset by sampling from this distribution. Furthermore, we scale this dataset to $[0, 1]^p$ by linear scaling with the range. This is done to comply with the domain of the test functions we will use further on.

We will cluster the data points into K clusters $\{C_1, \ldots, C_K\}$ with centers $\{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$ and use these as nodes. The centers are computed as the mean of the data points in that cluster. To do this, we investigate three different methods, namely k-means clustering, PCA-based clustering and a newly designed method called random split and combine clustering. In the following, we will use the words clustering and partitioning interchangeably.

3.1. K-means

The k-means method is one of the oldest and most widely used methods for clustering [15], [16]. The idea behind it is to minimize the within-cluster-sum of squares (SOS):

$$\min_{\{\mathbf{z}_1,\ldots,\mathbf{z}_K\}} SOS(\mathbf{z}_1, \ldots, \mathbf{z}_K), \quad SOS(\mathbf{z}_1, \ldots, \mathbf{z}_K) = \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{z}_{\mathrm{argmin}_k ||\mathbf{x}_i - \mathbf{z}_k||_2^2}||_2^2 \tag{18}$$

The implementation of the method consists of an initialization and an optimization phase, in which the initialization is done by random choosing k data points as cluster centers. After initialization, in each iteration step, first each data point is allocated to the nearest cluster center and then the cluster centers are recalculated as the mean of the data points assigned to it. For this, Euclidean distance is used, but other metrics might be used as well (however, for non-Euclidean norms it is not always guaranteed that the algorithm converges). When there are no more changes in the assignment of the data points to the clusters, the algorithm has converged and stops.

There are many extensions and improvements of the (initialization of the) algorithm. Some of them are very useful, such as using the triangle inequality to avoid unnecessary distance calculations [17], or initializing the method by PCA ([18], [19]). Other extensions worth mentioning are global methods ([20], [21], [22]) and low-rank approximations [23].

We will use the k-means++ method in this subsection, which has a special initialization as described in [24]. The idea behind it is that if the initial centroids are chosen (at random) such that the distances between the cluster centers are large, there is larger probability that the initial cluster centers are closer to the optimum.

Because the algorithm contains a random initialization and the objective function is non-convex, it can end in a local minimum, without guarantees how far this minimum is from the global optimum. Therefore, the algorithm is performed r times ($r > 1$) and the best result (with minimal SOS) is chosen. It might be that the algorithm does not always converge within the default number of steps as implemented in the MATLAB function `kmeans`. However, this will be ignored because in practice, most of the r executions converge and the chosen best result is a converged solution. Further onwards, we will refer to this method as KME.

3.2. PCA-based clustering

With this method [18], [19], one starts with a single large cluster, and in each step, the cluster with the largest average radius is split (which is equivalent to splitting the cluster with the largest sum of squares divided by the number of points in that cluster). Clustering by the diameter criterion is already performed by [25], but they split the cluster with the largest diameter. Division methods based on farthest centroids have been suggested by [26]. Here, we split such that the cutting plane goes through the old cluster center (center of mass) and is perpendicular to the largest principal component of the covariance matrix of the data as suggested by [19]. This continues until the largest average radius is smaller than some threshold $\sqrt{p \cdot \alpha \cdot \frac{N-1}{N}}$, or until a maximum number of clusters is attained. The threshold comes from the criterion that the relative difference between the weighted variance of the cluster centers and the sample variance of the data is smaller than α, which is a small number such as 0.05 or 0.01.

This method (referred to as PCA later on) is deterministic, which means that it will always give the same result for a specific data set. However, when a cluster is split, it cannot be merged again. This can be a drawback of the method. The merging of clusters will be explored in the next method.

3.3. Random split and combine clustering

This method (called RSC hereafter) is based on simulated annealing ([27], [28], [29], [30], [31]). In general, for simulated annealing one has to define an acceptance probability function, a temperature function and a cooling schedule, as well as a method to make local "moves" in the state space. Here, we define

a clustering method that is more restricted and requires only two parameters, namely the maximum number of clusters k_{max} and the maximum discrete time M. At each time step (or iteration) $t \leq M$, a cluster can be split (similar to PCA), or two clusters can be combined. In order to determine which two clusters are combined, one computes the new cluster center for all combinations of two clusters. The combination that is selected is the one that minimizes the maximum distance between the new center and the old centers.

The RSC algorithm is initialized with a single cluster and an obligatory split. At each iteration step, the algorithm chooses randomly between splitting and combining. If the number of clusters is k, the probability of splitting is given by

$$\begin{cases} P[\text{split at step } t] = \frac{1}{2}\left(1 + \exp(-\lambda t)\right) & \text{if } 1 < k < k_{max}\,, \\ P[\text{split at step } t] = 1 & \text{if } k = 1\,, \\ P[\text{split at step } t] = 0 & \text{if } k = k_{max}\,, \end{cases} \tag{19}$$

and P[combine at step t] = 1 - P[split at step t]. We determine the parameter λ by requiring that after $t = M/2$, the expectation for k would be k_{max} if there were no maximum imposed on k. In the calculation, we assume that at step 1, there is one cluster, and at step 2, a split is performed. To compute λ, we have to estimate the number of splits and the number of combinations. Because time is discrete, this estimate is a summation over time, which we will approximate by an integral. The calculation of λ is as follows:

$$\begin{aligned} 2 + \int_3^y \frac{1}{2}\left(1 + \exp(-\lambda x)\right) dx - \int_3^y \left(1 - \exp(-\lambda x)\right) dx = k_{max} \implies \\ \int_3^y \exp(-\lambda x) dx = k_{max} - 2 \implies \\ \frac{-1}{\lambda}\left[\exp(-\lambda y) - \exp(-3\lambda)\right] = k_{max} - 2. \end{aligned} \tag{20}$$

Equation (20) is solved by the built-in MATLAB numeric solver `vpasolve` for λ with initial guess 0.01.

When the algorithm finishes, then it might be that some clusters are non-convex and that part of the data points are not allocated to the nearest node. To correct for this, as a final step we reassign all the data points once to their nearest node. Note that this will always decrease the SOS.

In this method, we have removed the drawback of PCA, but in return, we got the randomness back. Furthermore, the time complexity of this method is dominated by the cost of combining, which increases quadratic with the number of clusters. For splitting, it is only linear in the number of clusters.

3.4. Calculation of weights

In all three methods, the weights of the nodes are determined by the number of points associated with that node, divided by the total number of data points. "Associated with" means here that this node is closest to the data point in the Euclidean metric.

3.5. Summary of the new methods

When input of an experiment is given as a dataset in p dimensions with N data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_i \in \mathbb{R}^{1\times p}$ also denoted by a matrix $X \in \mathbb{R}^{N\times p}$, we can use one of the proposed methods to compute nodes $\{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$ for a quadrature rule. The weights of the nodes are determined empirically. In this way, we can perform integration on the data or compute moments by using only the nodes instead of the complete data set. This is useful when computational or experimental resources are limited. With these methods, we lose the high degree-of-exactness of the Gauss quadrature rules, but it gives the opportunity to work with correlated data and it does not need a specified input distribution. Furthermore, the number of parameters of each of the methods is limited. For all three clustering methods, the maximum number of clusters k_{max} is a parameter. Further parameters are r (number of repetitions) for KME, α (variance threshold) for PCA and M (maximum discrete time) for RSC.

In the next section, we test how these clustering methods perform for collocation.

4. Results

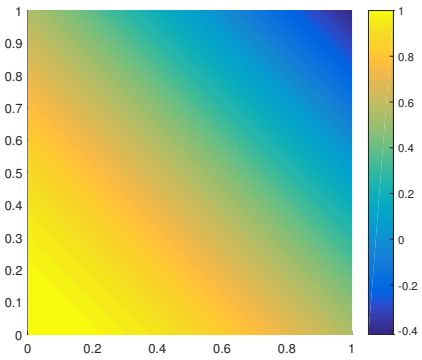
4.1. Genz' test functions

Genz [32] has developed several functions to test the accuracy of a cubature rule. Each function has its own characteristic which can be strengthened by the choice of a parameter $\mathbf{a}$. Another parameter, $\mathbf{u}$, can be used to shift the function. The functions are defined in p dimensions, in which $p \in \mathbb{N}$, on the domain $[0,1]^p$. In all tests, we will choose $a_i = 1$ for $i = 1, \ldots, p$. We will choose $u_i = 1/2$ for $i = 1, \ldots, p$ for all functions except for f_1, where we choose $u_1 = 0$. The definitions are given in Table 1.
We will test the methods by integrating the Genz' test functions over different data sets consisting of $N = 10^5$ samples, both through Monte-Carlo integration and integration based on the cluster points and weights of the different methods when applied to the data set. The difference between the two integrals is a measure for the (in)accuracy of the methods. The parameters in the algorithms are chosen as follows: PCA($\alpha = \epsilon$) (machine precision), RSC($M = 500$) and KME($r = 25$). This value of α is chosen such that the executions of the algorithm will attain the maximum number of clusters k_{max}. Furthermore, we compare the methods with partial Monte-Carlo (PMC). For this method, we take random k_{max} samples from the data set which all have the same weight.

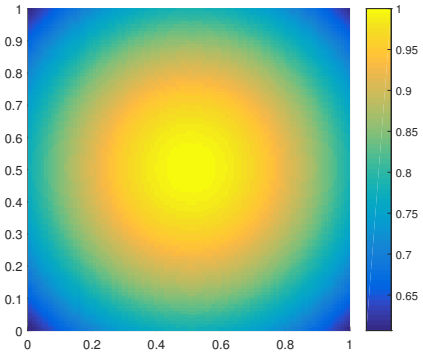
In Figure 2, the values for the Genz functions on the domain $[0,1]^2$ are visualized. Test function 2 and 4 look the same, but are different.

4.2. Data sets

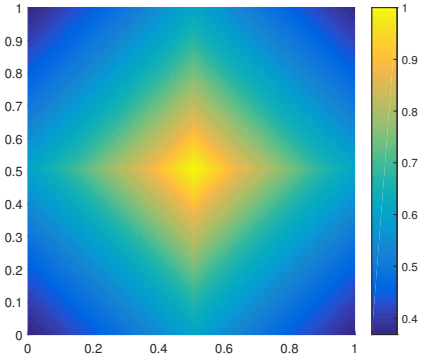
We will use three data sets with different levels of correlation to illustrate the methods. All sets consist of $N = 10^5$ samples drawn from a certain distribution. The dimension p is allowed to vary from 1 to 16. The first set will be a multivariate Gaussian distribution in p dimensions, the second set will be



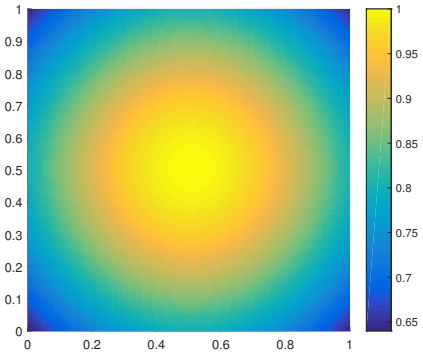
(a) Test function 1



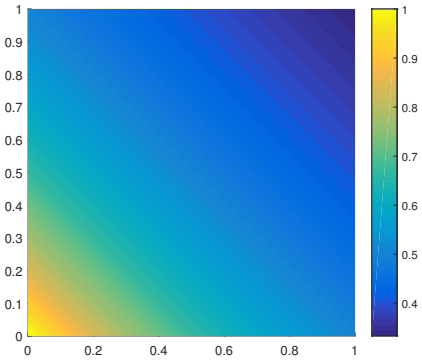
(b) Test function 2



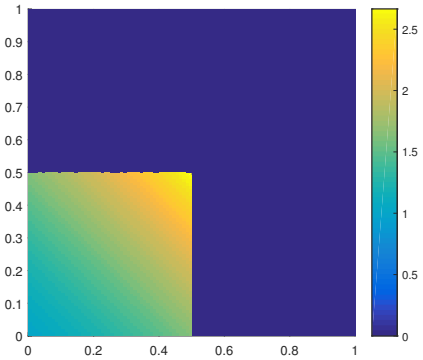
(c) Test function 3



(d) Test function 4



(e) Test function 5



(f) Test function 6

Figure 2: Genz test functions

Table 1: Definition of the Genz test functions

Nr	Characteristic	Function
1	Oscillatory	$f_1(\mathbf{x}) = \cos\left(2\pi u_1 + \sum_{i=1}^{p} a_i x_i\right)$
2	Gaussian peak	$f_2(\mathbf{x}) = \exp\left(-\sum_{i=1}^{p} a_i^2 (x_i - u_i)^2\right)$
3	C_0	$f_3(\mathbf{x}) = \exp\left(-\sum_{i=1}^{p} a_i \lvert x_i - u_i\rvert\right)$
4	Product peak	$f_4(\mathbf{x}) = \prod_{i=1}^{p} \left(a_i^{-2} + (x_i - u_i)^2\right)^{-1}$
5	Corner peak	$f_5(\mathbf{x}) = \left(1 + \sum_{i=1}^{p} a_i x_i\right)^{-p+1}$
6	Discontinuous	$f_6(\mathbf{x}) = \begin{cases} 0 & x_1 > u_1 \quad \text{or} \quad x_2 > u_2 \\ \exp\left(\sum_{i=1}^{p} a_i x_i\right) & \text{else} \end{cases}$

the uncorrelated beta distribution in p dimensions, and the third set will be an artificial data set which is strongly nonlinear correlated. After generation, they are rescaled to the domain $[0, 1]^p$ because the Genz' test functions are defined on the unit cube. Their parameters are given as follows.

The multivariate Gaussian distribution has zero mean, unit variance and correlations σ_{ij} between dimensions i and j given by

$$\sigma_{ij} = \frac{1}{|i - j| + 1}. \tag{21}$$

This is chosen such that neighboring dimensions have larger correlation than dimensions far apart. The beta distribution is the beta distribution with $\alpha = 2$ and $\beta = 5$ and its pdf is given by

$$f(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha - 1} (1 - x)^{\beta - 1}, \tag{22}$$

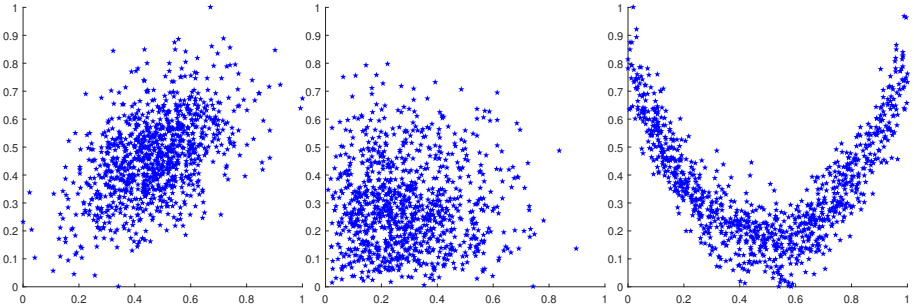
in which $B(\alpha, \beta)$ is the beta function.

The third and last distribution is given as

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix} = \begin{bmatrix} U(-2, 2) \\ X_1^2 \\ \vdots \\ X_1^p \end{bmatrix} + \sigma N(\mathbf{0}, I), \tag{23}$$

in which $U(-2, 2)$ is the uniform distribution on $[-2, 2]$, σ is chosen to be 0.5 and $N(\mathbf{0}, I)$ the multivariate standard normal distribution.

In Figure 3, we show $N = 10^3$ data points generated for $p = 2$ for the different test sets. From the figure, it is clear that these data sets have different types of correlations. The normal distributed data is weakly positive correlated, the beta distributed data is uncorrelated and the polynomial data is nonlinear and strongly correlated.



(a) Normal distributed data (b) Beta distributed data (c) Polynomial data

Figure 3: Visualization of the test sets for $p = 2$ and $N = 10^3$. The beta distributed data is uncorrelated, while the normal distributed data is weakly correlated and the polynomial data is strongly, nonlinearly correlated.

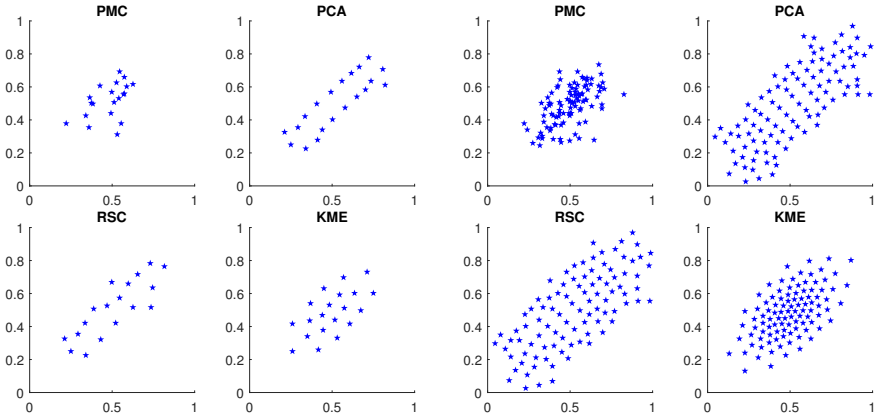
In Figure 4, the resulting partitionings for $k_{max} = 20$ and 100 for the different test sets with $N = 10^3$ are shown. One of the observations is that the PMC method takes most samples in dense regions, just as the KME method. In the latter, the spacing between the nodes is more evenly distributed in space. Both the PCA and RSC method also have nodes in less dense regions of the data set.

4.3. Tests

The tests of the methods will consist of integrating the test functions on each of the data sets by the nodes and weights of the proposed methods. We will perform Monte-Carlo integration on each of the data sets and each test function as a reference value. The data sets will be generated only once and reused. The output of each of the methods is the value of the integral of the test function when performed with the cluster points and weights. Not all results will be shown, but we will show representative examples. First, we compute the relative error as given by the absolute difference between the integral calculated by the cluster points and weights and the Monte-Carlo integral of the data, divided by the Monte-Carlo integral of the data. We do this for various values of p and test sets for a fixed maximum number of cluster points $k_{max} = 50$ to see how the error relates to dimension. The results are in Figure 5. In this figure, we want to show a trend which holds for all of the proposed methods: namely, that the relative error is in general lower for the correlated data sets, and especially for the polynomial data set, which is highly correlated. This indicates that the methods work especially well for correlated inputs, which is caused by the data being concentrated on or near a low-dimensional manifold rather than all of $\mathbb{R}^p$. Furthermore, even for $p = 16$, the results are favorable for the clustering-based methods.

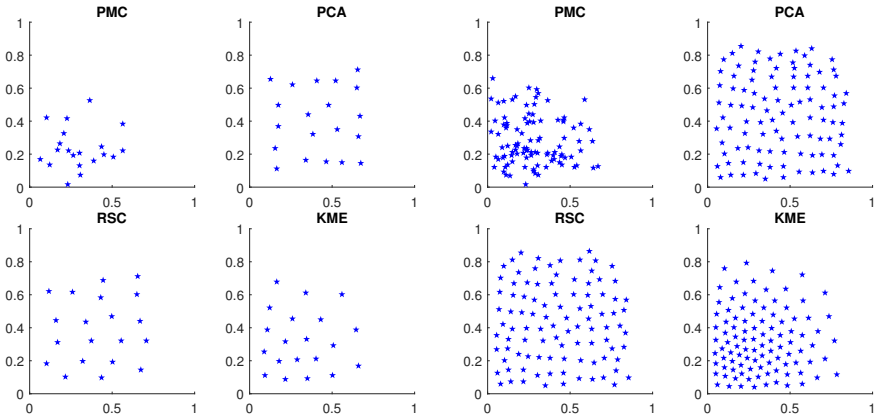
In Figure 6, the effect of increasing k_{max} is studied for the PCA-method for $p = 4$. The errors generally decrease with increasing k_{max}, although not very strongly.

Furthermore, the time it takes to compute the nodes are given in Figure 7. This is independent of the test function, because all test functions use the same nodes and weights. Increasing from 1 to 16 dimensions increases the computation time only by a factor 10, which means the overhead of computing nodes in higher dimensions is very small. PMC is the fastest method, but has less accurate results (note that PMC is not based on clustering). For RSC and KME, the choice of M and r influences the computation time. Their effects can be expected to be linear. For all four methods, the computation time is small, in particular when it is compared to the time-consuming, expensive evaluation of output functions one can encounter in applications.



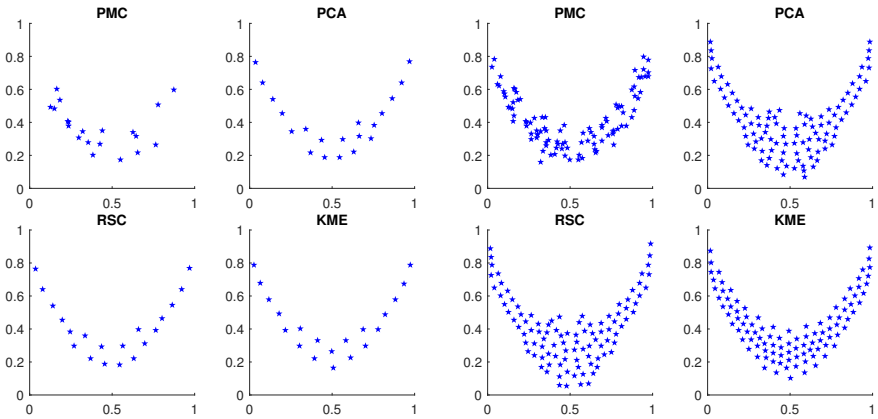
(a) Normal distributed data, $k_{max} = 20$

(b) Normal distributed data, $k_{max} = 100$



(c) Beta distributed data, $k_{max} = 20$

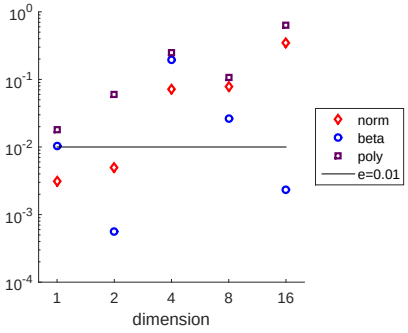
(d) Beta distributed data, $k_{max} = 100$



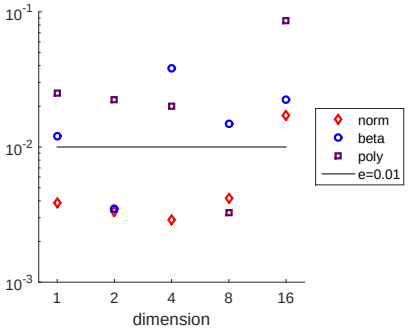
(e) Polynomial data, $k_{max} = 20$

(f) Polynomial data, $k_{max} = 100$

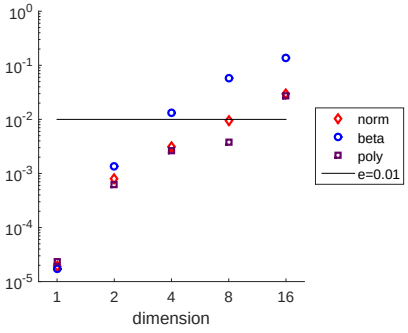
Figure 4: Visualization of the partitionings for $p = 2$. The general observation is that PMC and KME have most nodes in dense regions of the data, while PCA and RSC are more spread out over the domain of the data.



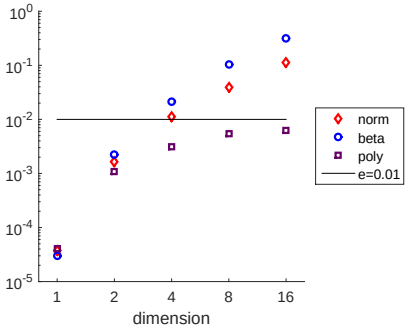
(a) PMC, test function 1



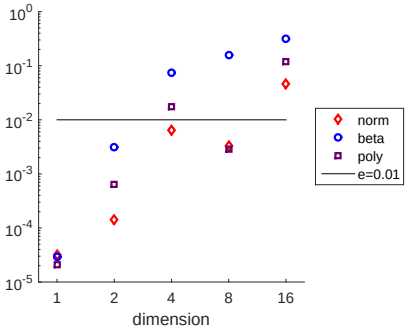
(b) PMC, test function 2



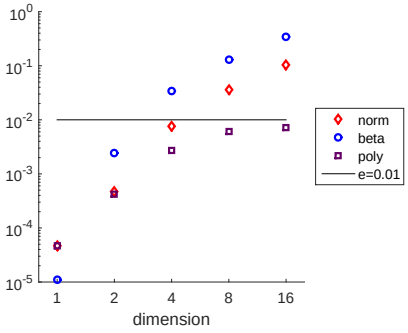
(c) PCA, test function 1



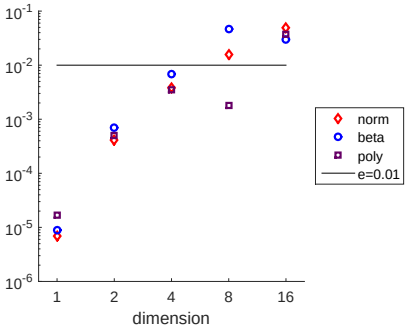
(d) PCA, test function 2



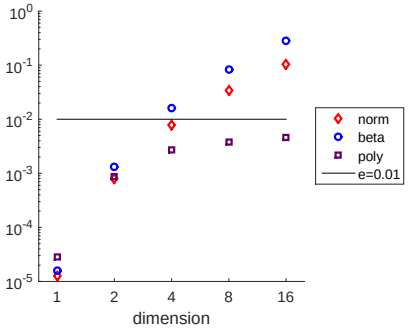
(e) RSC, test function 1



(f) RSC, test function 2

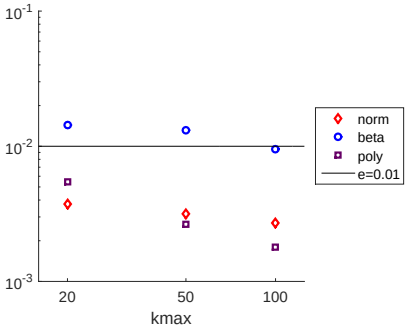


(g) KME, test function 1

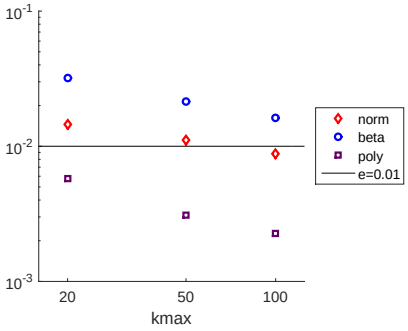


(h) KME, test function 2

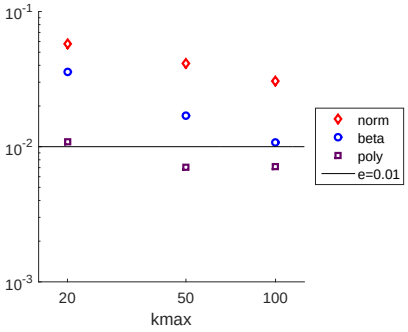
Figure 5: Relative error depending on dimension for different methods and data sets with $k_{max} = 50$.



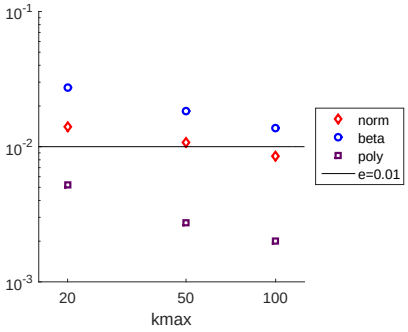
(a) Test function 1



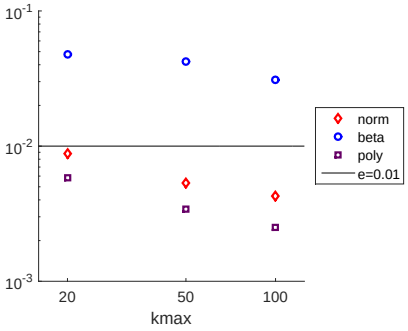
(b) Test function 2



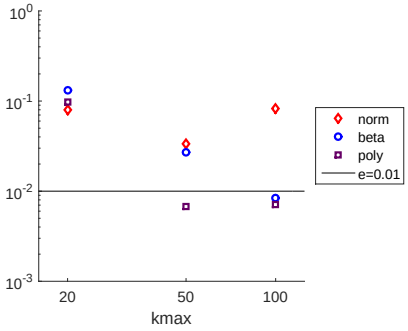
(c) Test function 3



(d) Test function 4

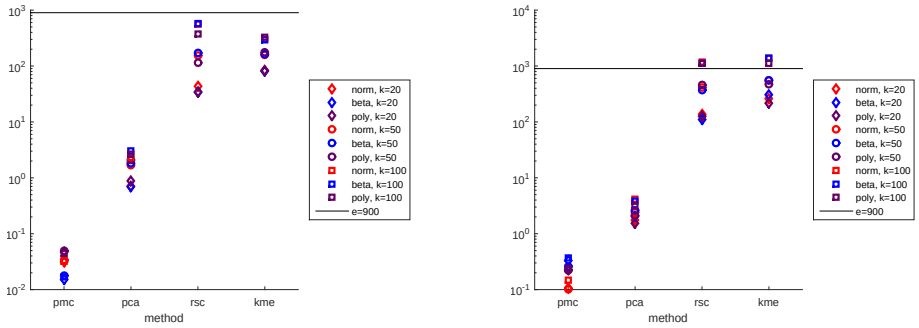


(e) Test function 5



(f) Test function 6

Figure 6: Relative error depending on k_{max} for PCA and the three data sets with $p = 4$. Increasing k_{max} has only a modest effect on the relative error.



(a) 1 dimension

(b) 16 dimensions

Figure 7: Time in seconds depending on method and data set for $p = 1$ and $p = 16$.

5. Comparison in 2D to SC with Gaussian quadrature

We have shown in the previous section the performance of the three methods, of which PCA seemed to perform best based on relative error and running time. In this section, we compare the results against "regular" stochastic collocation with a tensor grid of Gauss quadrature points as a benchmark result. To do so, we construct a two-dimensional correlated beta distribution with parameters $\alpha = 2$ and $\beta = 5$ which we can make uncorrelated by setting the correlation parameter ρ to zero. We can then use the Gauss-Jacobi quadrature points as nodes. For $l_{max} = 3$, with $n_l = 2^l + 1$, $l = 1, \ldots, l_{max}$, this leads to 81 grid points for the full grid for $p = 2$.

Because of the curse of dimension, it is hard to perform the comparison in higher dimensions because of the number of nodes needed. Furthermore, PCA works especially well for correlated data, but SC with a tensor grid is not designed for correlated data. Therefore, we use a modified version of the SC method when $\rho \neq 0$ by taking the uncorrelated quadrature points as nodes and determining the weights empirically as the fraction of data associated with each node (as described in Section 3.4). If the weight is zero, the node will be removed from the grid. This can be seen as a credulous or naive way of extending SC to correlated inputs, and we will denote it "SC".

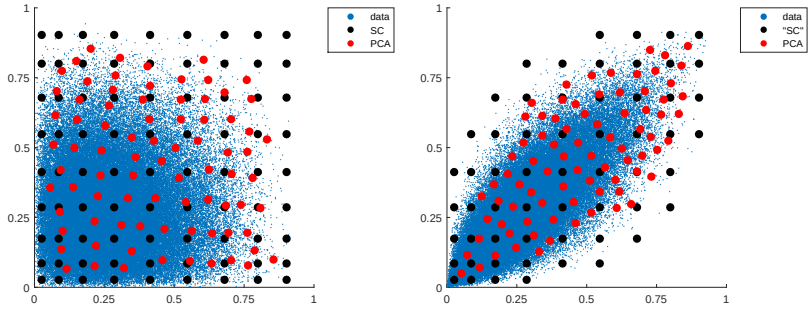
A correlated multivariate beta distribution can be constructed in the following way: start with a p-dimensional (here, $p = 2$) multivariate normal distribution with zero mean, unit variance and correlations given by ρ_{ij}, in which $\rho_{ii} = 1$ and $\rho_{ij,j\neq i}$ can be chosen freely in $[-1, 1]$. Its marginals are standard normal distributions. We take a large sample from this distribution and transform it into a correlated multivariate beta distribution by applying the cumulative distribution function of the normal distribution in each dimension to get values in $[0, 1]$ and then apply the inverse cumulative distribution function of the beta distribution, hence

$$x_\beta = CDF^{-1}(B(\alpha, \beta), CDF(N(0, 1), x_N)). \tag{24}$$

In this way, each dimension is mapped independently, but the output is correlated. In two dimensions, the correlation matrix is given by $\left[\begin{smallmatrix} 1 & \rho \\ \rho & 1 \end{smallmatrix}\right]$. For $\rho = 0$, there is no correlation and for $\rho = 1$ or $\rho = -1$, there is full correlation.

For $\rho = 0$ and $\rho = 0.8$, the results in terms of nodes are in Figure 8 for $k_{max} = 81$. The integration points are those resulting from PCA and SC, or its counterpart for $\rho = 0.8$. For visualization purposes, the test data set consisting of $N = 10^5$ points is plotted as well.

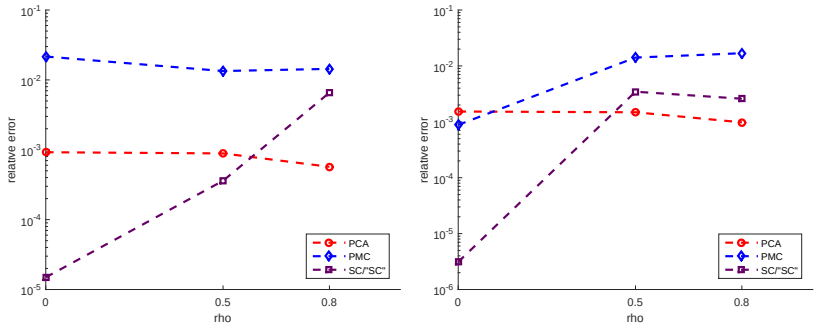
First, we compare the PCA and PMC method against a full grid consisting of Gauss-Jacobi quadrature points for this data set with $\rho = 0$. The PCA and PMC method are used with $k_{max} = 81$. After that, we choose ρ equal to 0.5 and 0.8, respectively. For these sets, we compare the PCA against the credulous SC and PMC method with $k_{max} = 81$. We again use the Genz test functions. The integrals are compared to the Monte-Carlo integration over the total data set and the results are in Figure 9. The error for the credulous SC method increases when ρ increases, which means this method does not work well (as expected).



(a) $\rho = 0$ (b) $\rho = 0.8$

Figure 8: Data and integration points.

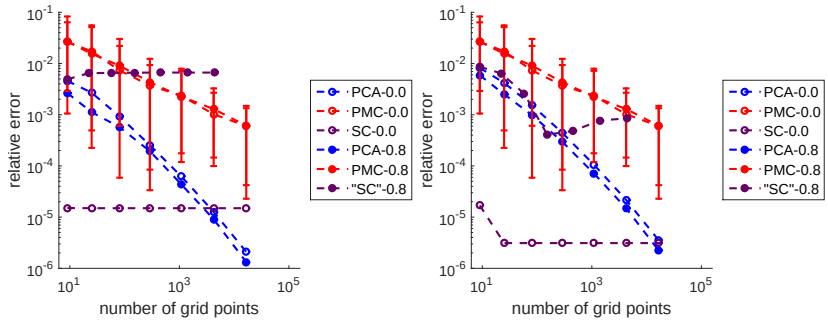
For PCA, the error is about the same for different values of ρ. The same holds for the PMC method, but with a considerably higher error.



(a) Test function 1 (b) Test function 2

Figure 9: Results for the different methods and correlation coefficients with $k = 89$ ($l = 3$).

We also looked at the convergence of the methods for increasing k_{max} for test function 1 and 2. The k_{max} of the full grid are given by $(2^l + 1)^2$, in which $l = 1, \ldots, 7$. To take the randomness of the PMC method into account, we have repeated the method 25 times and averaged the resulting relative error. In practice, this will often not be possible because of computational cost (hence we omitted it in the previous section). Both the average relative error and the range of the relative error are visualized. The results are in Figure 10. For uncorrelated data ($\rho = 0$), the SC method still works best. For correlated data, PCA is by far the best option when more than 25 nodes are available. Note that the nodes in both figures are the same, but the function evaluations differ.



(a) Test function 1 (b) Test function 2

Figure 10: Results for the different methods and correlation coefficients with varying k.

One of the results is that the full grid consisting of Gauss-Jacobi quadrature points does not seem to converge. The cause of this is that these grids integrate the test function over the domain with respect to the beta distribution, while the reference value is based on Monte-Carlo integration of samples of this beta distribution. When this is corrected, it is stable around 10^{-9} (from $l = 2$ onwards) and 10^{-14} (from $l = 3$ onwards), respectively. Furthermore, these figures indicate that the convergence of PCA is exponential in k_{max} for small k_{max}, while the PMC method converges as $O(k_{max}^{-1/2})$ as expected. When the number of grid points would be increased further, then for $l = 8$ more than half of the data points is included as a node, and for $l = 9$, all data points are included, because $n_9 = (2^9 + 1)^2 = 263169 > N = 10^5$. When all nodes are chosen as grid points, the error will be 0, because integration of the complete set was used for the reference values.

6. Conclusion

We have proposed a novel collocation method that employs clustering techniques, thereby successfully extending SC to the case of multivariate, correlated inputs. We have assessed the performance of this clustering-based collocation method using the Genz test functions as benchmark. Three clustering techniques were considered in this context, the KME, PCA and RSC techniques (as detailed in Section 3). All three techniques gave good results, especially in case of strongly correlated inputs (the "polynomial" data set, see Figures 5 and 6). We emphasize that no exact knowledge of the input distribution is needed for the clustering-based method proposed here, as a sample of input data is sufficient. Furthermore, for strongly correlated inputs the method showed good performance with input dimension up to 16. We hypothesize that the more strongly the inputs are correlated, the more the input data are concentrated on a low-dimensional manifold. This makes it possible to obtain a good representation of the input data with a relatively small number of cluster centers.

We compared the clustering-based method against Partial Monte-Carlo (PMC), which uses a random subsample of the full dataset as collocation nodes. The resulting nodes are mostly concentrated in regions of high density of the input probability distribution, with poor representation of the tails. As a result, the PMC method does not perform well. The clustering-based method, in particular the PCA and RSC variants, are much better at giving a good spread of the collocation nodes, see Figure 4. Concerning computational cost, the three considered clustering techniques compute the cluster points within 15 minutes on a standard modern PC. RSC and KME have almost the same running times, which can be adapted by changing the parameters M and r. PCA is the fastest, with a running time roughly between 1 and 10 seconds in our tests, see Figure 7. Overall, the computational cost of the clustering methods is small, and will be negligible compared to the computational cost of expensive model evaluations (involving e.g. CFD solvers) in applications.

Altogether, we would suggest to use the method based on principal component analysis (PCA) from the ones that we tested. This method is deterministic, fast to compute and yields collocation nodes that are well distributed over the input data set. PCA performed well on the tests with Genz functions.

The PCA method is also compared to regular SC using a Gaussian tensor grid and a simple adaptation thereof (for the case of correlated inputs), and there we have two clear observations. First, for uncorrelated inputs the regular SC works better, as was to be expected. We note that the errors when using PCA for uncorrelated data are also small (albeit not as small as regular SC). For correlated input variables, PCA works much better than the adapted SC (see Figure 10). Second, the PCA method has about the same performance for different gradations of correlation, in other words, the performance does not deteriorate as the correlation increases (see Figure 9).

The results in this study demonstrate that clustering-based collocation is a feasible and promising approach for UQ with correlated inputs. We intend to develop this approach further in the near future.

Acknowledgements

Very sadly, Jeroen Witteveen passed away unexpectedly in the early stages of the research reported here. His presence, inspiration and expertise are greatly missed. Jeroen was one of the initiators of the EUROS project, which includes the current work. This research, as part of the EUROS project, is supported by the Dutch Technology Foundation STW, which is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs.

References

[1] H. Bijl, D. Lucor, S. Mishra, C. Schwab, Uncertainty Quantification in Computational Fluid Dynamics, Vol. 92 of Lecture Notes in Computational Science and Engineering, Springer, 2013.

[2] R. W. Walters, L. Huyse, Uncertainty analysis for fluid mechanics with applications, Tech. rep., NASA (2002).

[3] J. A. Witteveen, H. Bijl, Efficient quantification of the effect of uncertainties in advection-diffusion problems using polynomial chaos, Numerical Heat Transfer, Part B: Fundamentals 53 (5) (2008) 437–465.

[4] J. A. Witteveen, S. Sarkar, H. Bijl, Modeling physical uncertainties in dynamic stall induced fluid–structure interaction of turbine blades using arbitrary polynomial chaos, Computers & structures 85 (11) (2007) 866–878.

[5] B. Yildirim, G. E. Karniadakis, Stochastic simulations of ocean waves: An uncertainty quantification study, Ocean Modelling 86 (2015) 15–35.

[6] D. Xiu, G. E. Karniadakis, The wiener–askey polynomial chaos for stochastic differential equations, SIAM journal on scientific computing 24 (2) (2002) 619–644.

[7] D. Xiu, J. S. Hesthaven, High-order collocation methods for differential equations with random inputs, SIAM Journal on Scientific Computing 27 (3) (2005) 1118–1139.

[8] R. G. Ghanem, P. D. Spanos, Stochastic finite elements: a spectral approach, Courier Corporation, 2003.

[9] O. P. Le Matre, O. M. Knio, Spectral methods for uncertainty quantification : with applications to computational fluid dynamics, Scientific computation, Springer, 2010.

[10] M. Eldred, J. Burkardt, Comparison of non-intrusive polynomial chaos and stochastic collocation methods for uncertainty quantification, AIAA paper 976 (2009) (2009) 1–20.

[11] M. Navarro, J. Witteveen, J. Blom, Stochastic collocation for correlated inputs, in: UNCECOMP 2015, 2015.

[12] S. A. Smolyak, Quadrature and interpolation formulas for tensor products of certain classes of functions, in: Dokl. Akad. Nauk SSSR, Vol. 4, 1963, p. 123.

[13] T. Gerstner, M. Griebel, Sparse grids, Encyclopedia of Quantitative Finance.

[14] C. W. Clenshaw, A. R. Curtis, A method for numerical integration on an automatic computer, Numerische Mathematik 2 (1) (1960) 197–205.

[15] H. Steinhaus, Sur la division des corps matériels en parties, Bulletin de l´Académie polonaise des sciences IV (12).

[16] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, 1967, pp. 281–297.

[17] C. Elkan, Using the triangle inequality to accelerate k-means, in: ICML, Vol. 3, 2003, pp. 147–153.

[18] C. Ding, X. He, K-means clustering via principal component analysis, in: Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 29.

[19] T. Su, J. Dy, A deterministic method for initializing k-means clustering, in: Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on, IEEE, 2004, pp. 784–786.

[20] A. Likas, N. Vlassis, J. J. Verbeek, The global k-means clustering algorithm, Pattern recognition 36 (2) (2003) 451–461.

[21] A. M. Bagirov, Modified global k-means algorithm for minimum sum-of-squares clustering problems, Pattern Recognition 41 (2008) 3192–3199.

[22] P. Hansen, E. Ngai, B. K. Cheung, N. Mladenovic, Analysis of global k-means, an incremental heuristic for minimum sum-of-squares clustering, Journal of classification 22 (2) (2005) 287–310.

[23] M. Cohen, S. Elder, C. Musco, C. Musco, M. Persu, Dimensionality reduction for k-means clustering and low rank approximation, arXiv preprint arXiv:1410.6801.

[24] D. Arthur, S. Vassilvitskii, K-means++: The advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[25] A. Guénoche, P. Hansen, B. Jaumard, Efficient algorithms for divisive hierarchical clustering with the diameter criterion, Journal of classification 8 (1) (1991) 5–30.

[26] H.-r. Fang, Y. Saad, Farthest centroids divisive clustering, in: Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on, IEEE, 2008, pp. 232–238.

[27] S. Kirkpatrick, M. P. Vecchi, et al., Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.

[28] R. W. Klein, R. C. Dubes, Experiments in projection and clustering by simulated annealing, Pattern Recognition 22 (2) (1989) 213–220.

[29] G. T. Perim, E. D. Wandekokem, F. M. Varejão, K-means initialization methods for improving clustering by simulated annealing, in: Advances in Artificial Intelligence–IBERAMIA 2008, Springer, 2008, pp. 133–142.

[30] K. Rose, Deterministic annealing for clustering, compression, classification, regression, and related optimization problems, Proceedings of the IEEE 86 (11) (1998) 2210–2239.

[31] S. Z. Selim, K. Alsultan, A simulated annealing algorithm for the clustering problem, Pattern recognition 24 (10) (1991) 1003–1008.

[32] A. Genz, Testing multidimensional integration routines, in: Proc. Of International Conference on Tools, Methods and Languages for Scientific and Engineering Computation, Elsevier North-Holland, Inc., New York, NY, USA, 1984, pp. 81–94.
URL http://dl.acm.org/citation.cfm?id=2837.2842