Methodology of Generating Output Space-filling Sample for Explicit Function

By

WANG Yi An, Andy
(2030005058)

A Final Year Project thesis (STAT4004)
submitted in partial fulfillment of the requirements
for the degree of

Bachelor of Science (Honours)
in Statistics

at

BNU-HKBU
UNITED INTERNATIONAL COLLEGE

November, 2023

DECLARATION

I hereby declare that all the work done in this Project is of my independent effort. I also certify that I have never submitted the idea and product of this Project for academic or employment credits.

WANG Yi An, Andy
(2030005058)

Date:

Methodology of Generating Output Space-filling Sample for Explicit Function

Wang Yi An, Andy

Faculty of Science and Technology

Abstract

The problem of sampling the range of a function given a multidimensional output is one of the most important invisible obstacles to some fields of statistics. It is very difficult to obtain input data that is purposefully generated so that the output data exhibits the properties we want. From one of the simpler cases, an input data that can make the range sampling meet our requirements can be considered as a representative point problem in an ideal state. The representative point problem plays an important role in the fields of uncertainty quantification, cluster analysis, Bayesian analysis, signal processing and statistical simulation. Therefore, this article proposes a new NFF (Normality Filling Filter) input sample generation algorithm, which can make the output of our target present the form we want. In this paper, we also compare the effect of kernel density estimation between its sampling method and representative points. Therefore, this paper urges statisticians to focus on the input-side sampling problem of output space filling in order to produce algorithm improvements that meet practical needs.

Keywords: Space-filling sample, Normality Filling Filtered(NFF), Range Space, OBSF, BOSF, OISF, Multi-normal Distribution, Cosine Similarity, Representative Points, Kernel Density Estimation.

Preface

In this thesis, you will have a chance read an really interesting and promising topic, which can be thought as a more advanced data sampling methodology from a generalized perspective.

This research artical will provide a brief summary about a really important concept induced from the uniform distribution, which is called space-filling sample. It gained an increasing attention of statistical field. And then point out the drawback of only taking the space-filling in input space. Then this article aims to provide a methodology to enhance the target of our observation.

Specifically, Representative Point (RP) provides a uni-dimensional way of substitution of the given probability function. It has already tended to a systematic branch of statistical sampling. And Design of Experiment (DoE) provides a simplified solution for multivariate input case start from anova model. But both methodology will likely to lose its efficiency if we apply a mapping to this sampling and aimed at exploring its behavior in the output space (range). Thus my novel methodology (NFF) can be regarded as a generalized algorithm of attaining this target.It can be defined on a multi-variate output circumstance for any dimensional cases.

This thesis will also introduce the main concepts followed with the precisely theoretical proof related to probability measure. This thesis has a exhaustive narration of the detailed methodology. Due to the time limitation, there is unavoidable to have some typos in it. Please understand, thank you∼

Acknowledgement

This is the start of acknowledgment, to be honest, it is really hard for me to start this writing in confront of this large size of space. Because it is really confusing for a statistics beginner to writing a whole content of thesis. But under the supervision and instruction of Associated Prof. A.M. Elsawah, I have the chance to start this challenging topic. But when I calm down, it is really a good time for me to have a retrospect of my undergraduate statistics learning time, which can be thought as a gift and thanks to all the people who helped me during this period.

First, I want to thank all the teachers who have helped me in the whole learning process, which contains convener of both major and elective courses. As a multicultural and vibrant community, UIC pursue a liberal arts education, which provided lots of choices to shape my future life. Teachers who play vital role in expressing the knowledge and spirit to me. During this process, I have the chance to enter the scientific research group.

Second, I want to thank my parents, who taking up the main part of financial support of my undergraduate study in UIC. Honestly, it is really difficult for lots of Chinese family to pay the tuition fee for four years.

Last but not least, I want to thank all the people who I met in my undergraduate years. All of you have been an integral part of my four years in college. Without you, my undergraduate years would be incomplete and I would lose a lot of happiness. Thank you for the fate that allowed us to meet in this wonderful time and accompany me to grow up.

Contents

1 Introduction **1**

1.1 Background description 2

1.2 Our contributions 3

2 Newly Defined Concepts and Assumptions **5**

2.1 Intuitive feeling of ISF and OSF 5

2.2 Boundary in Range Space 8

2.3 Output Space-filling Circumstances Discussion 9

2.3.1 Output Boundary Space-filling(OBSF) 9

2.3.2 Output Internal Space-filling(OISF) 10

2.4 Relationship among BOSF, OBSF and OISF 10

3 Closest Point Iteration **12**

3.1 Iteration Details 13

3.2 Invalid Circumstance 15

4 Generation Algorithm Illustration **16**

4.1 Generation Process Illustration 17

4.2 Functions Illustration 18

4.3 Algorithm Illustration 19

5 Illustrative Examples **23**

5.1 Uni-variate Output Case 23

5.1.1 Mixture Normal Distribution 25

5.1.2 χ^2 Distribution 25
5.1.3 Exponential Distribution 28
5.1.4 Beta Distribution 28
5.1.5 Gamma Distribution 28
5.1.6 Logistic Distribution 28
5.2 Multivariate Output Case 42
5.3 Conclusion and Future Work 44

List of Tables

3.1 First Closest Point iteration 13
3.2 Second Closest Point iteration 14
3.3 k^{th} Closest Point iteration 14

4.1 Denote of notations in pseudo-code 18
4.2 Constructed sub-functions. 19

5.1 OSF results for $f_1 \sim f_3$ (Part of) 40
5.2 OSF results for $f_4 \sim f_6$ (Part of) 41
5.3 Cosine Similarity of f_7 and f_8 43
4 OSF results for $f_1 \sim f_3$ (Part of) 51
5 OSF results for $f_4 \sim f_6$ (Part of) 52

List of Figures

2.1 Contrast of input space-filling(ISF) and output space-filling(OSF), 1000 samples (OBSF) 6

2.2 Contrast of input space-filling(ISF) and output space-filling(OSF), 1000 samples (OISF) 7

2.3 Contrast of input space-filling(ISF) and output space-filling(OSF) OBSF 8

2.4 Illustration of three types output of OSF. 10

3.1 Intuitive illustration of Lemma. 12

3.2 Invalid Approximation. 15

3.3 Valid Approximation. 15

4.1 Rough Normality Filling Filtered (NFF) Methodology Flow Chart. . 17

5.1 Output Space-filling for Mixture Normal. 26

5.2 Output Space-filling for $\chi^2(6)$. 27

5.3 Output Space-filling for $Exp(5)$. 29

5.4 Output Space-filling for $Beta(0.5, 0.5; x)$. 30

5.5 Output Space-filling for $\Gamma(x; 3, 2)$. 31

5.6 Output Space-filling for $Logit(x; 3, 2)$. 33

5.7 **Mixture Normal Estimation Comparison**(f_1).
Actual function(red), MERPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.9$. 34

5.8 χ^2 **Estimation Comparison**(f_2).
Actual function(red), MCRPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.9$. 35
5.9 **Exponential Distribution Estimation Comparison**(f_3).
Actual function(red), MCRPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.04$. 36
5.10 **Beta Distribution Estimation Comparison**(f_4).
Actual function(red), MCRPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.04$. 37
5.11 **Gamma Distribution Estimation Comparison**(f_4).
Actual function(red), MCRPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.5$. 38
5.12 **Logistic Distribution Estimation Comparison**(f_6).
Actual function(red), MCRPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.9$. 39
5.13 Space-filling in range space for f_7, $n = 100$ 43

Chapter 1

Introduction

Space-filling design (SFD) has gained increasing number of attention in the field of sampling and experimental design for its uniformity. This feature of space-filling design (SFD) can make sampling with large scale coverage, systematic and randomization. It means during the sampling time, it is more likely to sample different circumstance compared with pure random sampling. From 1979, when the Latin Hypercube Design (LHD) first proposed, which provided a solid background on the following development. After that, the other kinds of space-filling, such as Minimax Design, Maximin Design and Maximin Latin Hypercube Design were proposed. Compared to the Ordinal Space-filling design, relevant researches on exploring the property of Output space-filling is rare. It is understandable to think that the variation of function or the unpredictable shape of range can be a main barrier to attain this target. In lots of cases, especially for some field, controlling the behaviors of output from the perspective of contolling the input has significant impact on the goodness of sampling. In this means, we developed a generalized method which is called Normality Filling Filtered (NFF) methodology to generate Output Space-filling (OSF).

1.1 Background description

Despite the potential wide usage of generating Output Space-filling Sample (OSFD) design, existing researches are mainly focus on proposing ideal theoretical definitions and requirements of output space-filling sample. The concept of it was firstly proposed in the process of solving acoustic meta-surfaces via using space-filling points (Krishna, Craig, and Shi, et al.(2022)). Following Motivated from the research on potential energy from crystal structures (Krishna, Tran, Huang, Ramprasad, and Joseph.(2022)), the detailed definition of output space-filling sample was firstly proposed in the perspective of maximize the minimum distance between two points (Wang, General, Kalidindi and Joseph. (2023)). And the objective function is

$$\max_{y \in \mathbf{y}} \inf_{y_i \in f(D_n)} d_Y(y, y_i) \tag{1.1}$$

where $d_Y(\bullet)$ means the distance in range space. It is used to maximize the minimum distance between any two points in the output space. Obviously it is hard to solve it from perspective of optimization, so mini-max perturbation algorithm (MP Algorithm) was proposed corresponded to the Mini-max distance (Krishna, Craig, and Shi, et al.(2022)). It is a dynamic optimization process to determine the distance between theoretical coordinate and actual iterative points. When the distance less than the threshold Δ, the iteration will stop, which can be expressed as

$$y^* = \arg\min_{y \in \mathbf{y}} ||x^* - f(y)|| \leq \Delta \tag{1.2}$$

The algorithm was enhanced from the perspective of Bayesian optimization with expected improvement (EI), and Brownian Motion. They also provide case studies to illustrates that the input space-filling design (ISFD) are usually not have the property of output space-filling (In FIG 1) in (Wang, General, Kalidindi and Joseph. (2023)). And the promoted and created research is to proposed idea on the Input-response space-filling designs, which is a enhanced and balanced version of design(Lu,

Anderson-Cook.(2021)). And it taken both the input and output into consideration

$$D = \{d_{ij} = \alpha d_{ij,X} + (1-\alpha)d_{ij,Y}\}, \qquad \alpha \in [0,1] \tag{1.3}$$

where $d_{ij,X}$ and $d_{ij,Y}$ corresponding to the ISF and OSF. Thus, we can found that related researches of output space-filling (OSF) are mostly gather in interpreting relative theorems and case studies, but till now, the novels of systematically illustration of generating OSF is hardly to be found. From our perspective, there are some hardships we often in confront to when considering generating such kinds of samples.First, the format of function can various a lot. Sometimes it may besimilar to a black-box, forming an insufficient output information can be obtained. The function also may not be reversible, especially for those multivariate function, which become a block of forming a direct generation if we use common inverse transform method.In addition, if we consider the generation process as an iterative optimization problem, it will be impractical since it does not have a effective manipulating value in generalized cases. If the input size is really large, the points in output space are asymptotically approximate to space covering, but it is a rather high cost of computational resources. Furthermore, the distinguish criteria of evaluating output space-filling(OSF) property are ambiguous to some extent. Due to the unpredictable and abstract shape of range, and non-convex properties in many cases, evaluating of space-filling is really difficult, especially for non-regular output space. Thus, to generate OSF sample, we need to provide proper assumptions and clear definitions to make it clear.

1.2 Our contributions

In the previous discussion, we have already discussed the Input Space-filling(ISF), which is let X to be space-filling. We have already mentioned due to the highly nondeterministic property of output space.Thus, this paper will focus on providing exactly conception definition and the algorithm of generating OSF sample in a specific range. We name it as Normality Filling Filtered (NFF) method. NFF made it possible for us to attain the target of OSF sample generation. It is a method which can be

generalized easily, which is a simulation way.

We proposed a novel technique to generate the approximate output space-filling input sample. This proposed adaptive methodology comes from a specific function or just a simple mapping. Thus, we will detailed discuss the main concepts and provide some detailed information to validate the effectiveness of sampling. To have a better illustration, we updated some theories which was widely used in other publications and provide relevant illustration here. For Section 2, we are discussing some situations that may be confront in different circumstances of OSF in range space. It will cover the roughly definition of boundary of range and the boundary we would like to detect. We will also have a detailed analysis on the relationship between Output Internal Space-filling (OISF), Output Boundary Space-filling (OBSF) and Bounded Output Space-filling (BOSF), distinguishing there similarities and differences. For Section 3, a detailed process of generation will be proposed. By using the novel method, which we call it as Normality filling filtered (NFF) methodology. It's effectiveness of space-filling is theoretically proved via the Banach-Caccioppoli Theorem. In Section 4, we will provide several numerical case studies related to the simplist circumstance of selecting Representative Points(RPs) and use kernel density estimation(KDE) to examine the performance in reality. Finally, we will summary the whole paper and draw conclusion.

Under our examine, it is also okay to do not know the exact form, so it is quite adaptive. A key feature for the result generated by this method is highly customized. By changing the pre-defined parameters, we can obtain different kinds of samples to meet the demand in reality. This methodology can also be applied in the non-convex output space.

Chapter 2

Newly Defined Concepts and Assumptions

2.1 Intuitive feeling of ISF and OSF

This figure shows a comparison between ISF and OSF. The first row is the ISF circumstance, while the second row shows the OSF circumstance. From the comparison of four sub-figures. It is simple find that the patterns of space-filling ususally have inconsistent behavior. (FIG 1) In this means, generate a sample such that the output can filling the range space under a given mapping f is of great important thing need to be done.

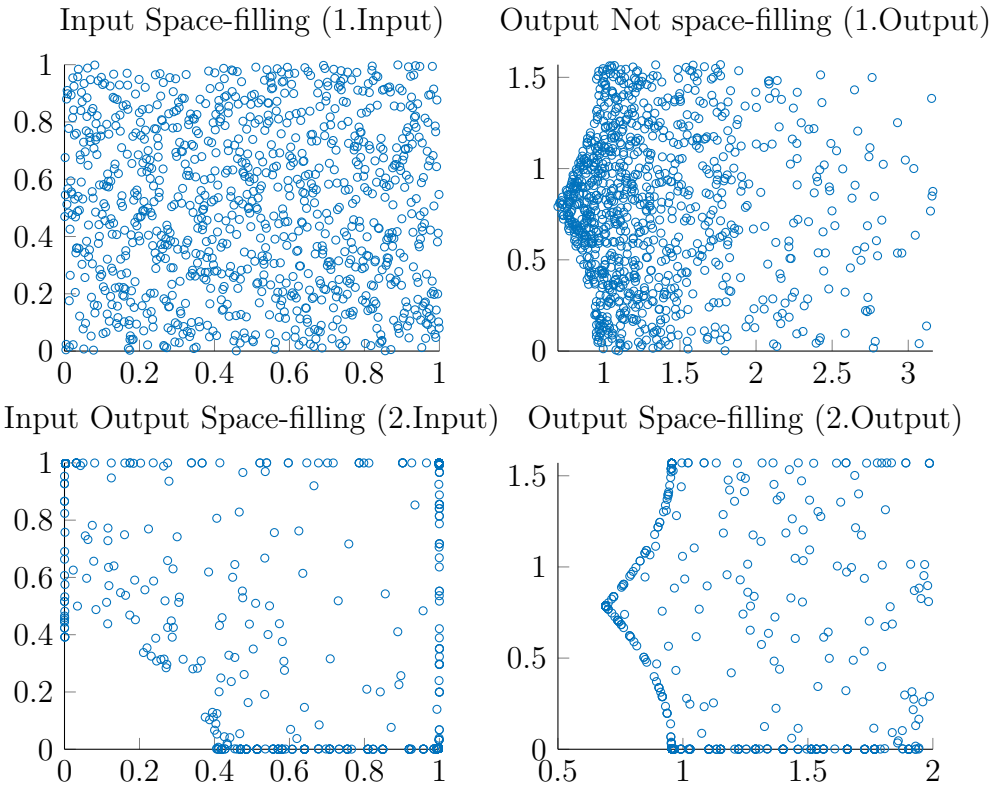

Figure 2.1: Contrast of input space-filling(ISF) and output space-filling(OSF), 1000 samples (OBSF)

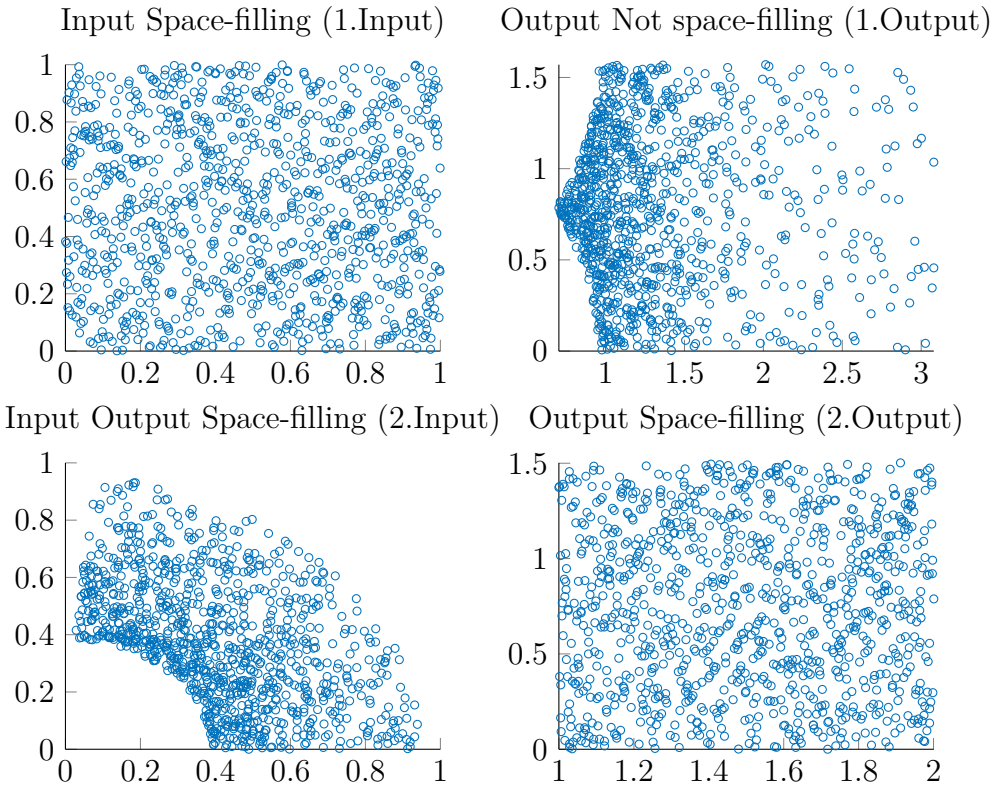

Figure 2.2: Contrast of input space-filling(ISF) and output space-filling(OSF), 1000 samples (OISF)

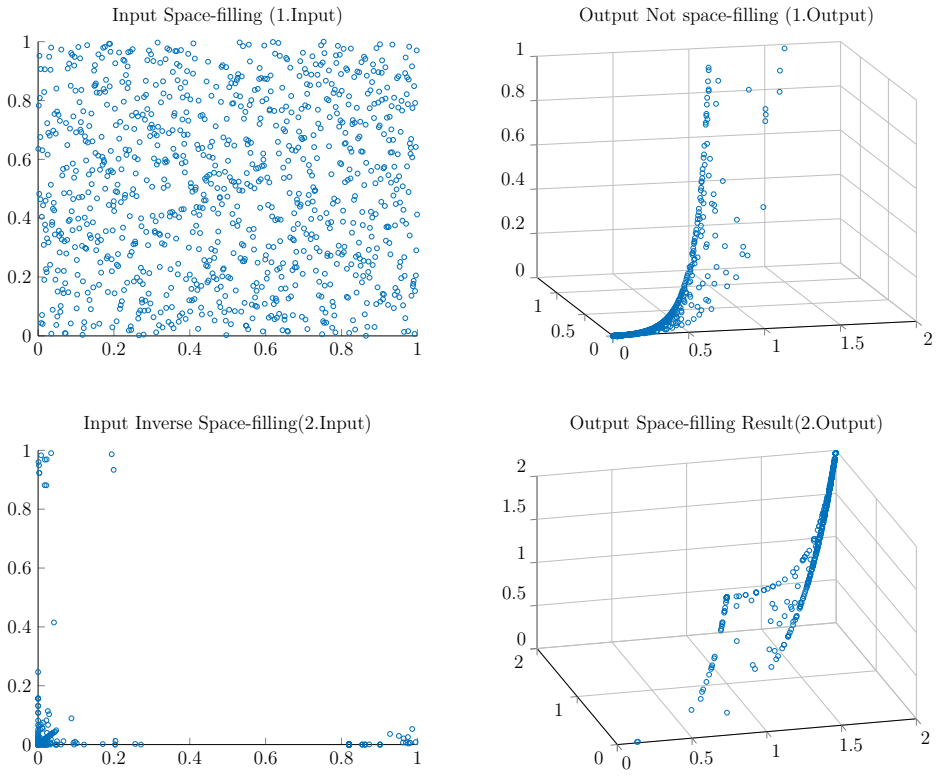

Figure 2.3: Contrast of input space-filling(ISF) and output space-filling(OSF) OBSF

2.2 Boundary in Range Space

There will be different shape of range spaces among different functions. Boundaries of the range shape is of great importance because they determine the evaluation of result. They can be divided into two types: hard boundary and soft boundary.

Definition 2.2.1. Hard Boundary and Soft Boundary

- Hard Boundary means the boundary which has a clear division between range and non-range space.

- Soft Boundary means the range space is boundless in any range directions. That is $\lim_{x\to\infty} f(x) = \infty$ or $\lim_{x\to c} f(x) = \infty$

To avoid the influence of ∞, we assume that the f is continuous and bounded in its domain and it will also continue to be used in the following parts. More specifically, we can also apply the range of region as we want to be obtained. This bounded both in range and input sample is called Bounded Output- Space-filling(BOSF).

2.3 Output Space-filling Circumstances Discussion

Since f is a bounded function, the range of it must also has its boundary. To meet a demand of space-filling in a range space will lead to two different directions, which are Output Boundary Space-filling(OBSF) and Output Internal Space-filling (OIS).

2.3.1 Output Boundary Space-filling(OBSF)

Intuitively, OBSF can be regarded as the space-filling with boundary. It is simple to be obtained via a preliminary specific sample as filter. Denote the range space of function f as $R(f)$ and the range of filter sample as $R(s)$. Assume that the point $p_i \in R(f)$ and $q_j \in R(s)$. Our target is

$$\forall\ q_j \in R(s),\ \exists p_i \in R(f),\ \text{s.t. min}\ d(q_j, p_k)\ \ \text{iff}\ k = i \tag{2.1}$$

where $d(\bullet)$ is the distance between two points. And the input of such points p_i is what we wanted. Since there may be some points $p \in R(s)\backslash R(f)$, which makes the point has nearest distance gathered in the boundary of $R(f)$, as the scatter pattern of fig 2.1and fig 2.3 shows.
In this case, most of the range points are gathered on the boundary of range and it cause a kind of rarefaction inside the range. This kind of problem will be dealt in next part – Output Internal Space-filling.

2.3.2 Output Internal Space-filling(OISF)

Intuitively, for output space-filling case, there is another case as the points all located inside the range space. However, due to the non-regular shape of range and the limitation of who paper, we cannot discuss it here. If no special explanation, the OSF mentioned in this passage are BOSF(Bounded Output Space-filling) or OBSF(Output Boundary Space-filling).

2.4 Relationship among BOSF, OBSF and OISF

In previous parts, we have already mentioned BOSF, OBSF and OISF. It is of great necessity to illustrate what is the difference between them in real circumstances.

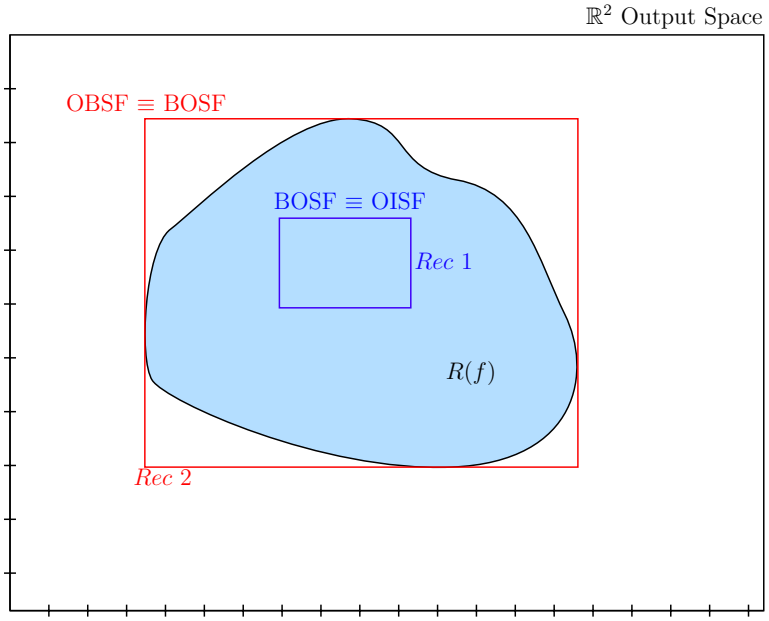

Figure 2.4: Illustration of three types output of OSF.

FIG 4 is an example of a circumstance of $\mathbb{R}^2$ output space. *Rec* 1 and *Rec* 2 represents two different circumstances of target range we assigned prior to generation of the OSF. The target range, accompanied with the range of function f (i.e. $R(f)$), have deeply influences the efficiency and property of behaviors in the output space.

If we expand the size of rectangle area completely inside range space, which is similar to *Rec* 1. In this circumstance, we will obtain a BOSF result from range space perspective. Meanwhile, it is also a OISF result of the target region. Then, we keep expanding it, until there are some vertices out of the $R(f)$, we will obtain a result which output space behavior has joint impact of range space $R(f)$ and bounded we settled for output space. If we keep expanding it, with every edge only have one common point to the boundary of $R(f)$, which is *Rec* 2. From this size of target and keep expanding, the points will increasingly gathers in the boundary of range space $R(f)$. Under this condition, the boundary of OBSF will exactly equal the pattern of boundary of $R(f)$. The bounded we set prior before generation will lose its effect. Thus, the behavior of OBSF in range space of this case is exact same to BOSF. If we extend it to higher dimensional case, the square region will turn to cubic or hypercube. And the circumstances will be more complicated but similar with the behavior in $\mathbb{R}^2$. In practice, it is of great important to choose proper bound of range space to meet the demand.

In following two chapters, we will introduce two most important part in the process of Normality Filling Filtered (NFF) methodology. The "Normality" means that we will apply multivariate normal distribution in making candidate points. And "Filling Filtered" means that we will apply the space-filling sample as a filter to select the candidate points

Chapter 3

Closest Point Iteration

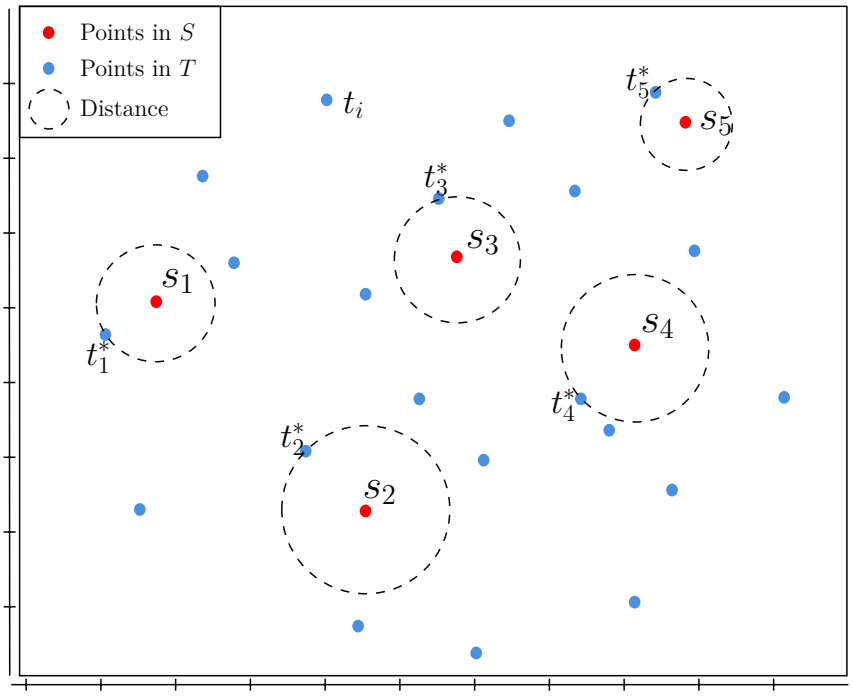

Figure 3.1: Intuitive illustration of Lemma.

Denote $R(f)$ as the range of function $f(\bullet)$. Then we will use Closest-point substitution method. That is by using the closest distance from a point set which distribution is known, we can easily find the substitution point set with same distribution.From previous discussion, we have already obtained raw input data.We need to

select some from the known parts to meet different distribution demands.

> Given two point sets S and T, the components of $S = \{s_1, s_2, ..., s_m\}$ and $T = \{t_1, t_2, ..., t_n\}$ have the same bound, and $|T| = n >> m = |S|$. We take the point t_i from T which has the minimum distance to a specific s_j. Rename it as t_j^* and find all of it to construct a new set $T^* = \{t_1^*, t_2^*, ..., t_m^*\}$. Then the pattern of T^* will approximate to the pattern in S, and denote it as $T^* \overset{\mathcal{D}}{\approx} S$.

Obviously, it is not enough to have only the nearest point for only one times, since there are a quite long distance between the filtered point and candidate points. In this means, we need to iteratively generate the candidates.
The chosen candidates are listed in the table below, the points $s_j, j = 1, 2, ..., N$ are the filtered points (target of Closest Point Iteration). And t_{ij}^* represents the i^{th} selection which target is point s_j in the first iteration (only one * in this process). Based on this the table below shows the relationships exhaustively.

3.1 Iteration Details

First Iteration

filter points	s_1	s_2	s_3	s_4	...	s_N
First time selection	t_{11}^*	t_{12}^*	t_{13}^*	t_{14}^*	...	t_{1N}^*
Second time selection	t_{21}^*	t_{22}^*	t_{23}^*	t_{24}^*	...	t_{2N}^*
Third time selection	t_{31}^*	t_{32}^*	t_{33}^*	t_{34}^*	...	t_{3N}^*
...	...	...	...	...	...	...
k^{th} time selection	t_{k1}^*	t_{k2}^*	t_{k3}^*	t_{k4}^*	...	t_{kN}^*
...	...	...	...	...	...	...
n^{th} time selection	t_{n1}^*	t_{n2}^*	t_{n3}^*	t_{n4}^*	...	t_{nN}^*

Table 3.1: First Closest Point iteration

And then, we choose the points which has the minimum distance among the closest

candidates, and denote it as t_i^{**}

$$t_j^{**} = \left\{ t_{ij}^{*} | \min d(t_{ij}^{*}, s_j), i = 1, .., n; j = 1, ..., N \right\} \tag{3.1}$$

Second Iteration

And then, we choose the points which has the minimum distance among the closest

filter points	s_1	s_2	s_3	s_4	...	s_N
First time selection	t_{11}^{**}	t_{12}^{**}	t_{13}^{**}	t_{14}^{**}	...	t_{1N}^{**}
Second time selection	t_{21}^{**}	t_{22}^{**}	t_{23}^{**}	t_{24}^{**}	...	t_{2N}^{**}
Third time selection	t_{31}^{**}	t_{32}^{**}	t_{33}^{**}	t_{34}^{**}	...	t_{3N}^{**}
...	...	...	...	...	...	...
k^{th} time selection	t_{k1}^{**}	t_{k2}^{**}	t_{k3}^{**}	t_{k4}^{**}	...	t_{kN}^{**}
...	...	...	...	...	...	...
n^{th} time selection	t_{n1}^{**}	t_{n2}^{**}	t_{n3}^{**}	t_{n4}^{**}	...	t_{nN}^{**}

Table 3.2: Second Closest Point iteration

candidates, and denote it as t_i^{***}

$$t_j^{***} = \left\{ t_{ij}^{**} | \min d(t_{ij}^{**}, s_j), i = 1, .., n; j = 1, ..., N \right\} \tag{3.2}$$

k^{th} Iteration

And then, we choose the points which has the minimum distance among the closest

filter points	s_1	s_2	s_3	s_4	...	s_N
First time selection	t_{11}^{k*}	t_{12}^{k*}	t_{13}^{k*}	t_{14}^{k*}	...	t_{1N}^{k*}
Second time selection	t_{21}^{k*}	t_{22}^{k*}	t_{23}^{k*}	t_{24}^{k*}	...	t_{2N}^{k*}
Third time selection	t_{31}^{k*}	t_{32}^{k*}	t_{33}^{k*}	t_{34}^{k*}	...	t_{3N}^{k*}
...	...	...	...	...	...	...
k^{th} time selection	t_{k1}^{k*}	t_{k2}^{k*}	t_{k3}^{k*}	t_{k4}^{k*}	...	t_{kN}^{k*}
...	...	...	...	...	...	...
n^{th} time selection	t_{n1}^{k*}	t_{n2}^{k*}	t_{n3}^{k*}	t_{n4}^{k*}	...	t_{nN}^{k*}

Table 3.3: k^{th} Closest Point iteration

candidates, and denote it as $t_i^{(k+1)*}$

$$t_i^{(k+1)*} = \left\{ t_{ij}^{k*} | \min d(t_{ij}^{k*}, s_j), i = 1, .., n; j = 1, ..., N \right\} \tag{3.3}$$

After performing lots of times of iteration (k is large), the distance between the candidate points and filter point is approximate to zero. Thus, we can think that the selected candidate points can approximately have the same distribution as the filtered point. And it finally attain the target of OSF with same pattern.

3.2 Invalid Circumstance

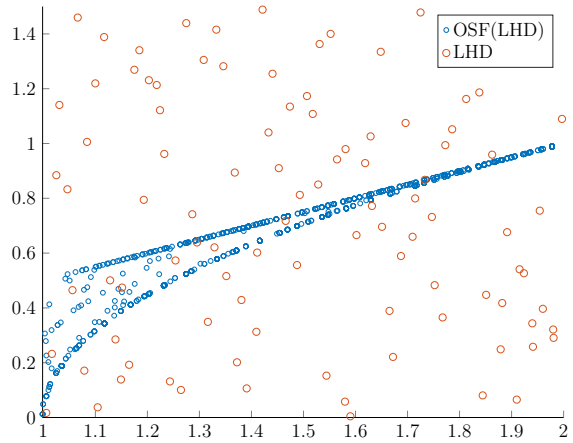

Figure 3.2: Invalid Approximation.

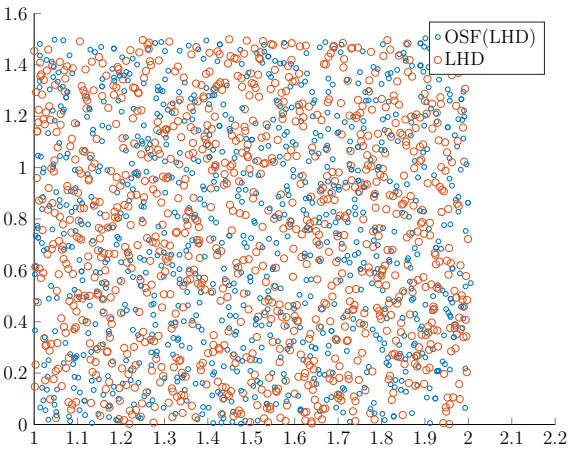

Figure 3.3: Valid Approximation.

As we mentioned previous, if range boundary has significant influence on the output space(Figure 3.2), which is the circumstance of BOSF. In the following narrative, it is the circumstance we cannot determine the goodness of result of NFF. We need to reset the bound we will observe and re-generate the sample. The cosine similarity can only deal with the circumstance of OISF (Figure 3.3).

Chapter 4

Generation Algorithm Illustration

The generation algorithm is named as NFF method. It constructed by several pure simulations sections.
To elaborate, in the following discussion, let us define two vector space $X \in \mathbb{R}^m$ and $Y \in \mathbb{R}^n$. There is a mapping f defined on X and map to Y, which is

$$f : X \longrightarrow Y \text{ or } X \xrightarrow{f} Y$$

Equivalently, a bounded function f has m inputs and n outputs (i.e. $Y = [y_1, y_2, ..., y_n] = f(x_1, x_2, .., x_m) = f(X) \equiv \mathbb{R}^m \to \mathbb{R}^n$). For all the independent variables in this function are bounded, non-negative and non-piecewise, that is $x_i \in [LB_i, UB_i]$. Since its output is a coordinate $(y_1, y_2, ..., y_n)$ and it can be decomposed into several marginal functions.

$$\begin{aligned} y_1 &= \tilde{f}_1(x_1, x_2, ..., x_m) \\ y_2 &= \tilde{f}_2(x_1, x_2, ..., x_m) \\ \dots &= \quad \dots \\ y_i &= \tilde{f}_i(x_1, x_2, .., x_m) \\ \dots &= \quad \dots \\ y_n &= \tilde{f}_n(x_1, x_2, ..., x_m) \\ f(\bullet) &= [\tilde{f}_1(\bullet), \tilde{f}_2(\bullet), ..., \tilde{f}_n(\bullet)] \end{aligned} \tag{4.1}$$

In the following sections, the operations are actually did in every marginal functions.

4.1 Generation Process Illustration

The sketch process of this generation is shown as this flow chart indicates.

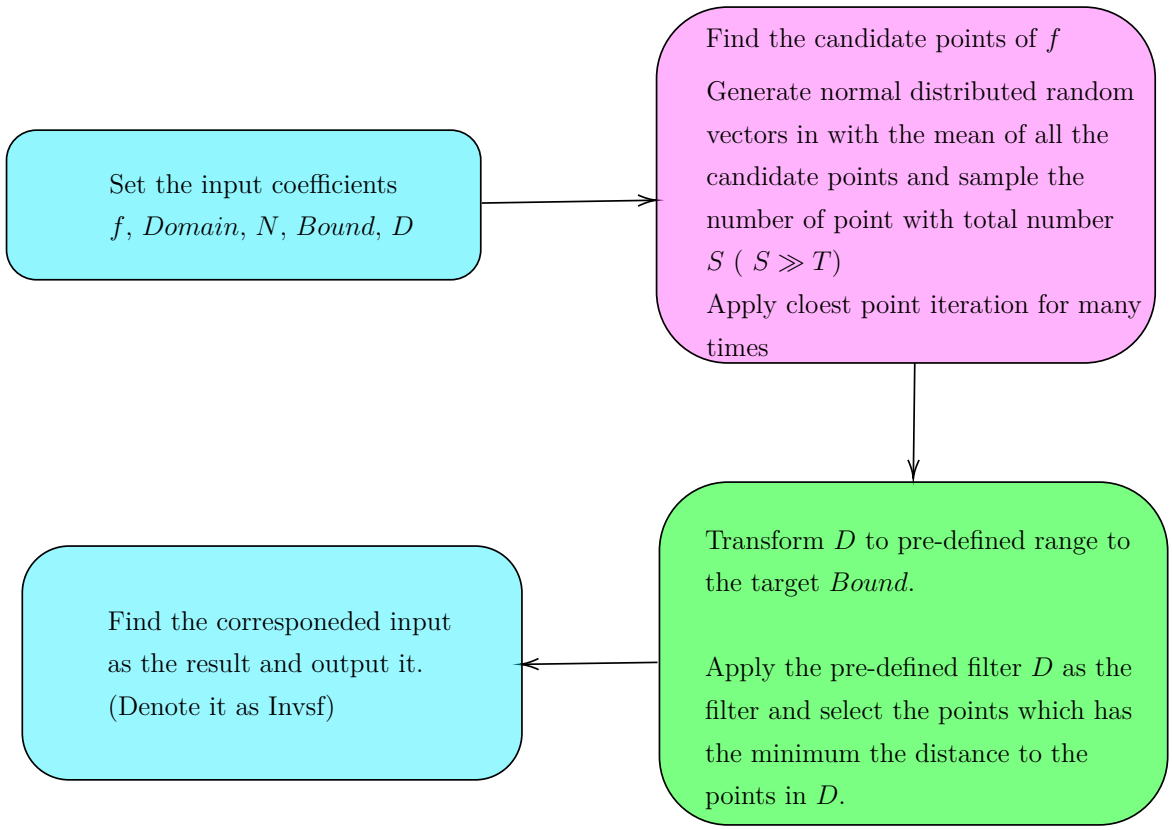

Figure 4.1: Rough Normality Filling Filtered (NFF) Methodology Flow Chart.

During this process, the balance of efficiency and effect is of great vital. On the one hand, since the theorem only holds on a significant higher dense of candidates, thus we need to compute lot of candidate points. On the other hand, we will be trapped in the low generation speed. Regarding this circumstance, and after lots of trials, we set the relationship between the actual generation points and candidate

points as

$$C = 10^m \times N \tag{4.2}$$

Where C is the total number of candidate points. a is the number of independent input variables. And N is the total number of points we want to generate. This adjustment of adding an exponential term is regarding the curse of dimensionality, which is based on the exponential increase on sample size when the dimension of sampling space is increasing.

4.2 Functions Illustration

From previous narrative, we need to set several parameters in this function.

Table 4.1: Denote of notations in pseudo-code

Notations	Meaning
f	Input function
Domain	Input domain matrix should be written as $Domain = \begin{bmatrix} LB_1 & UB_1 \\ LB_2 & UB_2 \\ ... & ... \\ LB_m & UB_m \end{bmatrix}$ for m input domain.
N	The number of points we want to obtain for the result sample.
Bound	Predefined observable region space for output space should be written as $Bound = \begin{bmatrix} LBB_1 & UBB_1 \\ LBB_2 & UBB_2 \\ ... & ... \\ LBB_n & UBB_n \end{bmatrix}$ *LBB*: Lower Boundary Bound. *UBB*: Lower Boundary Bound.
D	The filter sample for generation. It should have the same size of N.
InvSF	The output space-filling result.

In later narratives, we will also use the same notation when writing algorithm.

maxfind	Find the mean value which located in the extreme points and max/min point .
ndim_Normal	Generate multi-dimensional normal distribution which takes the coordinate of points candidates founded in **maxfind**.
filter	Remove all the input vectors which are outside of the domain.
Min_dis_filter	Select the point which satisfy the minimum distance to every pre-defined points in D.

Table 4.2: Constructed sub-functions.

In this section, we will exhaustively illustrate the realization of the whole process. The whole computational process can be divided into several sub-functions: In the **Max_find** function, due to various property of functions, there are several kinds of points are need to be taken into consideration. The first type is extreme value point, and it often indicates the local maximum/ minimum for the (marginal) function $f(x)$ or $\tilde{f}_i(x)$. And it may become the boundary point from the perspective of range space. The second type is the mid-point of the domain. It works well especially the function doesn't have extreme value point. And the type is maximum/ minimum value point. These points may be overlapped with some of points in the first type. Thus, the constructed algorithm is based on these three types of points.
In the next function **ndim_Normal**, we apply multi-variate normal distribution to generate candidate points. The mean of every normal probability density function is from all the coordinates of candidate points. And the covariance matrix we set it as $\Sigma = 10 \times diag\{1, ..., 1\}_{m \times m}$ (this is also internal defined value, and it is changeable). The **filter** function is used as the remove of any input points which are outside the domain. And function **Mis_dis_filter** is used to select the candidate points which have the minimum distance with the pre-defind points in D. This function can make the sample flexibly applied to satisfy demands in a variety circumstances.

4.3 Algorithm Illustration

Algorithm 1 Body Code

Input f, g, $Domain$, N, $Bound$, D.
Initialize $Invsf$.

Obtain the number of rows in $Domain$, and denote it as a.
Find all the candidate mean value as MU via **maxfind** function.
Find candidates of input points via **ndim_Normal** function.
Cancel the points which are out of the domain via **filter** function.
Use **Min_dis_filter** function to select the proper points (minimum distance).

Repeat the parts insides two lines for n times ▷ Closest Point Interation.
Output $Invsf$.

Algorithm 2 Max_find function.

Input coefficients f, g and $Domain$.
Initialize matrix M.
if f is uni-dimensional function **then**
Find the mid value of f, which is $f(\frac{LB+UB}{2})$.
Find all the extreme value point of f in all the marginal function $f(x)$.
else
Find all the extreme value point of f in all the marginal function $\tilde{f}_i(x)$.
Find all the mid-value of f in all marginal function $\tilde{f}_i(x)$, which is $f(\frac{LB_i+UB_i}{2})$ and add its input to matrix M.
end if

Record all the types of Inputs in matrix M and output it.

Algorithm 3 ndim_normal function.

Input k, mu. ▷ k is the total number of point will be generated.
Initialize $Input$. ▷ mu is the mean.
Obtain the size of mu as a $a \times b$ matrix.
for j from 1 to a **do**
Generate the Covariance matrix $Covmat$.
▷ we usually use$Covmat$ as a $diag\{10, ..., 10\}_{b\times b}$ matrix.
Initialize $i = 0$.
while $i < k$ **do** ▷ Iteratively generate every input.
Generate the normal distributed random vector number r, which has a mean $mu(j, :)$ and variance. $Covmat$.
$i = i + 1$.
end while
Collect all he values of r by matrix $Input$. And then output it.
end for

Algorithm 4 filter function.

Input two matrices $Input$ and $Domain$.
Initialize the Out as $a-$row, $b-$column matrix.
for i from 1 to a **do**
for j from 1 to b **do**
Cancel all the points in $Input$, which has at least one value out of the corresponded domain.
And denote the new $Input$ as Out
end for
end for
Output matrix Out.

Algorithm 5 Mis_dis_filter function.

Input variables D_1, f_D_1, $Bound$, D ▷ D_1: the sampling matrix input.
Initialize matrix Inv. ▷ f_D_1: the sampling matrix output.
Find the size of D as $n \times s$ ▷ n-row, s-column matrix.
▷ D is the filtered matrix.

for i from 1 to s **do**
Transform the range of values D to $Bound$.
Denote the transformed range of matrix D as D'.
end for
Find all the items in f_D_1 which has the closest distance to every point in D'.
Add all the points which are satisfying the condition of last line into matrix Inv and output it.

Chapter 5

Illustrative Examples

In this section, we will introduce the different kinds of example functions to give a illustration on the efficiency of NFF Methodology. I will briefly introduce the two circumstances of function – uni-variate output distribution and multi-variate output functions.

5.1 Uni-variate Output Case

In statistical narration, the distribution of a data is of great importance to be investigated. In this means, the representative point (RP) was widely used in many aspects. And Monte Carlo RPs (MCRPs), quasi-Monte Carlo RPs (QMCRPs) and minimum mean square error RPs (MMSERPs) are the most frequently used recommended classes of RPs. These classes of RPs are exhausted studied and plenty of ways for generating them given several continuous distributions. For instance, Fang et al.(2014) compared the performance of these three classes of RPs in statistical inference for the standard normal distribution. Sequentially, Jiang et al.(2015) explored the performance of QMCRPs and MMSERPs for the standard arcsine distribution. Recently, Xu et al.(2021) tested the performance of these three classes of RPs for the standard exponential distribution. Then, Li et al.(2022) investigated the performance of these three classes of RPs for a mixture of two normal distributions. Finally, Elsawah et al.(2023) proposed a new methodology for generating the Minimum Energy

RPs (MERPs) in the applying the formula of electric potential energy to generate RPs.

As the simplest case of OSF, I applied the OSF sampling result to six specific distributions to evaluate its performance of kernel density estimation (KDE). The background of such comparison is that even if the RPs is generated as input itself, they still reflect the property of the specific distribution. Thus, such comparison is feasible and practical.

In the KDE process, I took it as the sample input. That is given n RPs $R = \{r_1, ..., r_n\}$ from a specified continuous distribution with pdf $f(x)$, its corresponded kernel function (we use normal kernel) is

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - r_i) = \frac{1}{nh\sqrt{2\pi}}\sum_{i=1}^{n} e^{-\frac{1}{2}(\frac{x-r_i}{h})^2} \tag{5.1}$$

In the following uni-variate distributions, the selection of different distribution taking lots of properties into consideration. In 6.1.1, this mixture normal distribution is taking the multiple extreme value point into consideration. χ^2, Gamma, and Exponential is skewed distribution. The logistic function is symmetric distributed. The Beta Distribution $B(0.5, 0.5; x)$ has the unlimited upper bound. By applying NFF methodology to these distributions, and apply the KDE on it, we can evaluate the goodness of the application of OSF on the field of RPs.

Thus, we talk about the parameter setting in the OSF sample generation in the uni-variate circumstance. There are several cases we may in confront in reality. Parameter setting for generations can be separated into several circumstances. Assume that we want to observe the range space is

- If both x and y are bounded, and $X \in [LB, UB]$. We take Bound as $[y_{min}, y_{max}] \subset [LB, UB]$=Bound. And LB, UB should not have great difference than its corresponded y_{min} and y_{max}.
- If only y is bounded. Then we can add a constrain which will not have significant influence on bound in the range space.
- If only X is bounded. Then add a constrain in domain which will not make the

value of $f(x)$ exploded.

The definition of max-min distance (Johnson, Moore, and Ylvisaker 1990) is Suppose there are two points x_i and x_j, we need to find the minimum distance between the image of them and then maximize it in its range space $R(f)$.

$$\max_{R(f)} \min_{i,j} d(f(x_i), f(x_j)) \tag{5.2}$$

In the following computational results, we will use mini-max design as a filter to conduct the whole process.

5.1.1 Mixture Normal Distribution

Denote $X \sim N(\mu, \sigma^2)$ as the pdf of normal distribution with mean μ and variance σ^2. Then the mixture normal distribution is $X \sim \sum_{i=1}^{p} \alpha_i N(\mu_i, \sigma_i^2)$, and $\sum_{i=1}^{p} \alpha_i = 1$, where $x \in (-\infty, +\infty)$. We denote the mixed two normal distribution example is

$$\begin{aligned} f_1(x) &= 0.7 \times N(8, 10) + 0.3 \times N(1.5, 1) \\ &= \frac{0.7}{\sqrt{20\pi}} e^{-\frac{1}{2}(\frac{x-8}{\sqrt{10}})^2} + \frac{0.3}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-1.5)^2} \end{aligned} \tag{5.3}$$

The function value has a property of $\lim_{x \to \pm\infty} f_1(x) = 0$. And the range is bounded, that is $y \in [0, \max\{f_1(8), f_1(1.5)\}]$. And we set the $[y_{min}, y_{max}] \subset [LB, UB]$=Bound. In this example, we will choose Domain $= [-20, 40]$ and Bound $= [0, 0.14]$.

5.1.2 χ^2 Distribution

The second function is χ^2 distribution, which has a expression as

$$f_2(x) = \chi^2(6) = \frac{1}{2^3\Gamma(3)} x^3 e^{-\frac{x}{2}} = \frac{x^3 e^{-\frac{x}{2}}}{16}, x \in [0, +\infty) \tag{5.4}$$

Similarly, in this example, we will choose Domain $= [0, 30]$ and Bound $= [0, 0.14]$ as the input of .

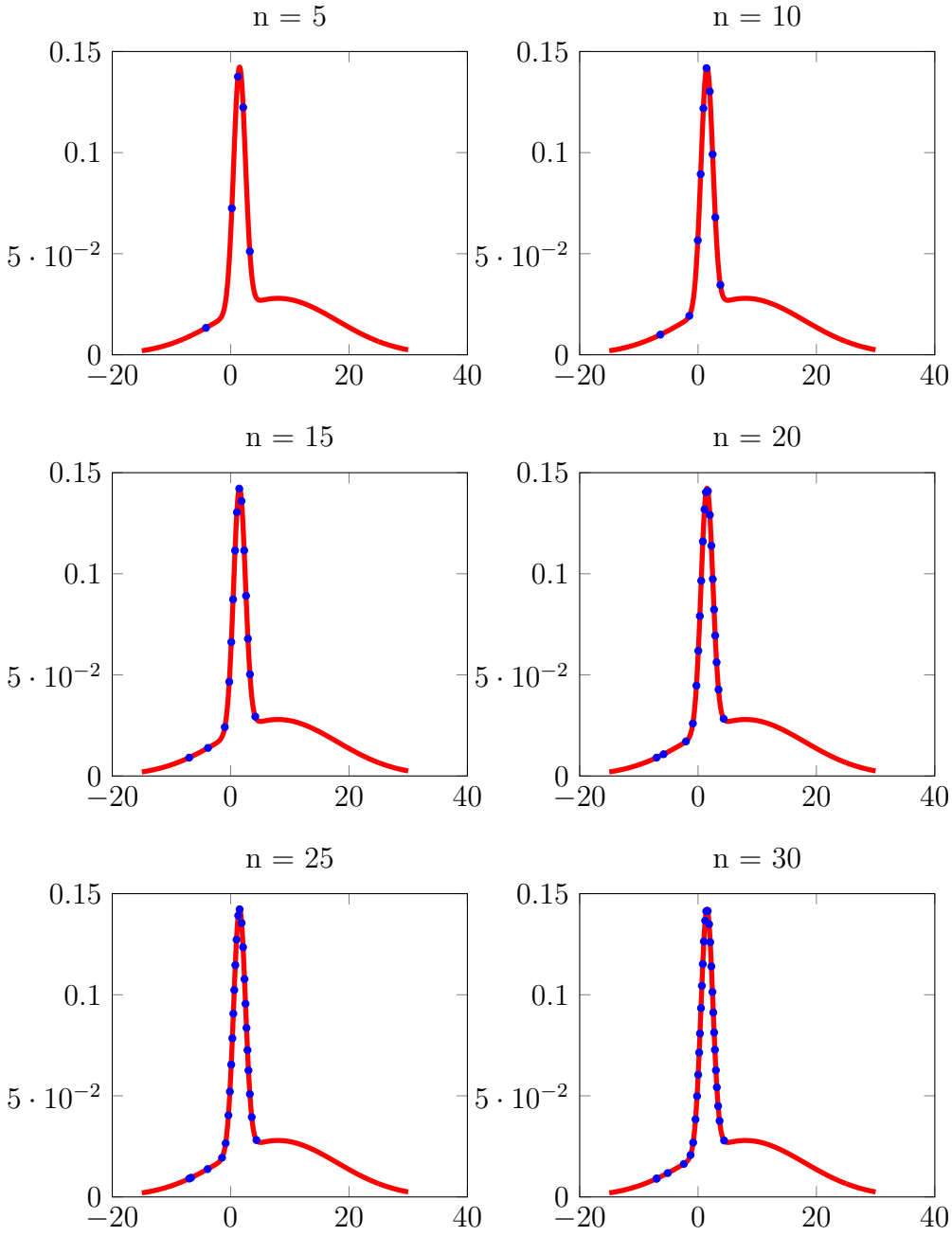

Figure 5.1: Output Space-filling for Mixture Normal.

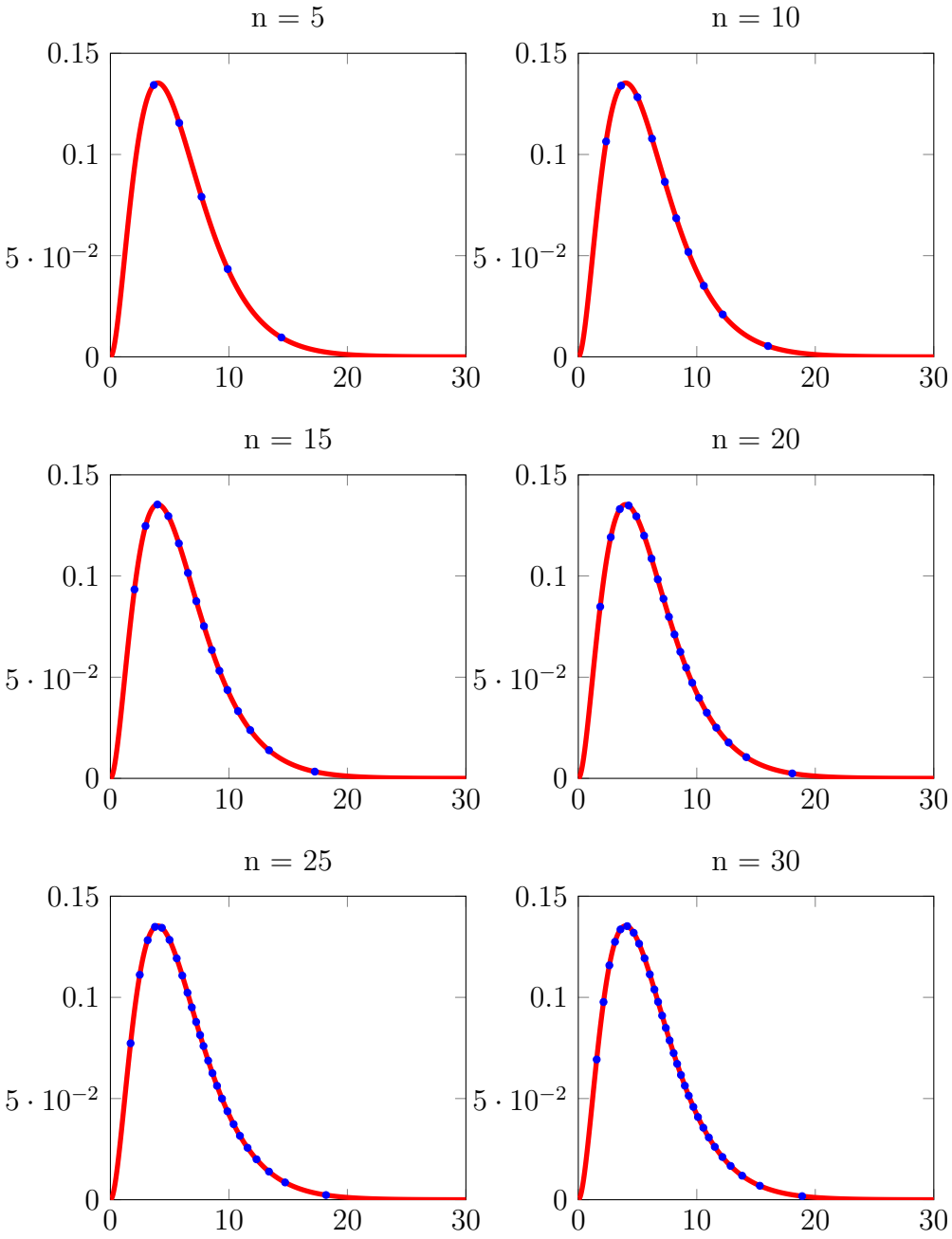

Figure 5.2: Output Space-filling for $\chi^2(6)$.

5.1.3 Exponential Distribution

The third function is exponential distribution, which has a expression as

$$f_3(x) = 5e^{-5x} \tag{5.5}$$

Similarly, in this example, we will choose Domain = $[0, 5]$ and Bound = $[0, 5]$.

5.1.4 Beta Distribution

The forth function is beta distribution, which has a expression as

$$f_4(x) = B(0.5, 0.5; x) = \frac{x^{0.5-1}(1-x)^{0.5-1}}{B(0.5, 0.5)} = \frac{1}{\pi\sqrt{x(1-x)}}, x \in (0, 1) \tag{5.6}$$

Similarly, in this example, we will choose Domain = $[0.01, 0.99]$ and Bound = $[0.5, 3.5]$. Since, for beta distribution $f_4(x)$, $\lim\limits_{x\to 0^+} f_4(x) = \lim\limits_{x\to 1^-} f_4(x) = +\infty$. Thus, we add a small value to make the value of $f_4(x)$ not blow up.

5.1.5 Gamma Distribution

The fifth function is Gamma distribution, which has an expression as

$$f_5(x) = Gamma(3, 2; x) = \frac{8}{\Gamma(3)} x^2 e^{-2x} \tag{5.7}$$

For f_5, we will choose Domain = $[0, 7]$ and Bound = $[0, 0.45]$.

5.1.6 Logistic Distribution

The sixth function is logistic distribution, which has an expression as

$$f_6(x) = \frac{e^{-\frac{(x-2)}{3}}}{3(1 + e^{-\frac{(x-2)}{3}})^2} \tag{5.8}$$

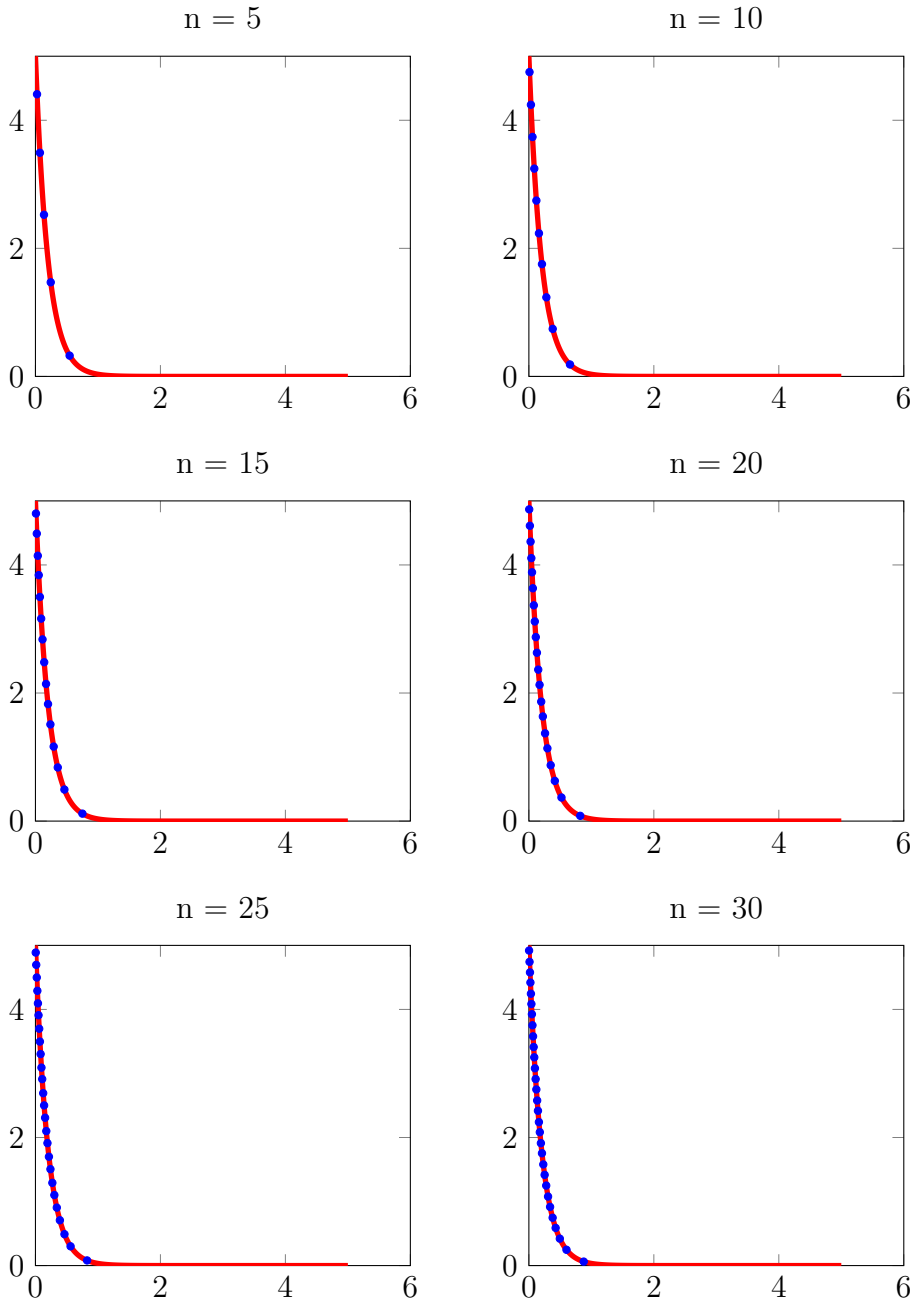

Figure 5.3: Output Space-filling for $Exp(5)$.

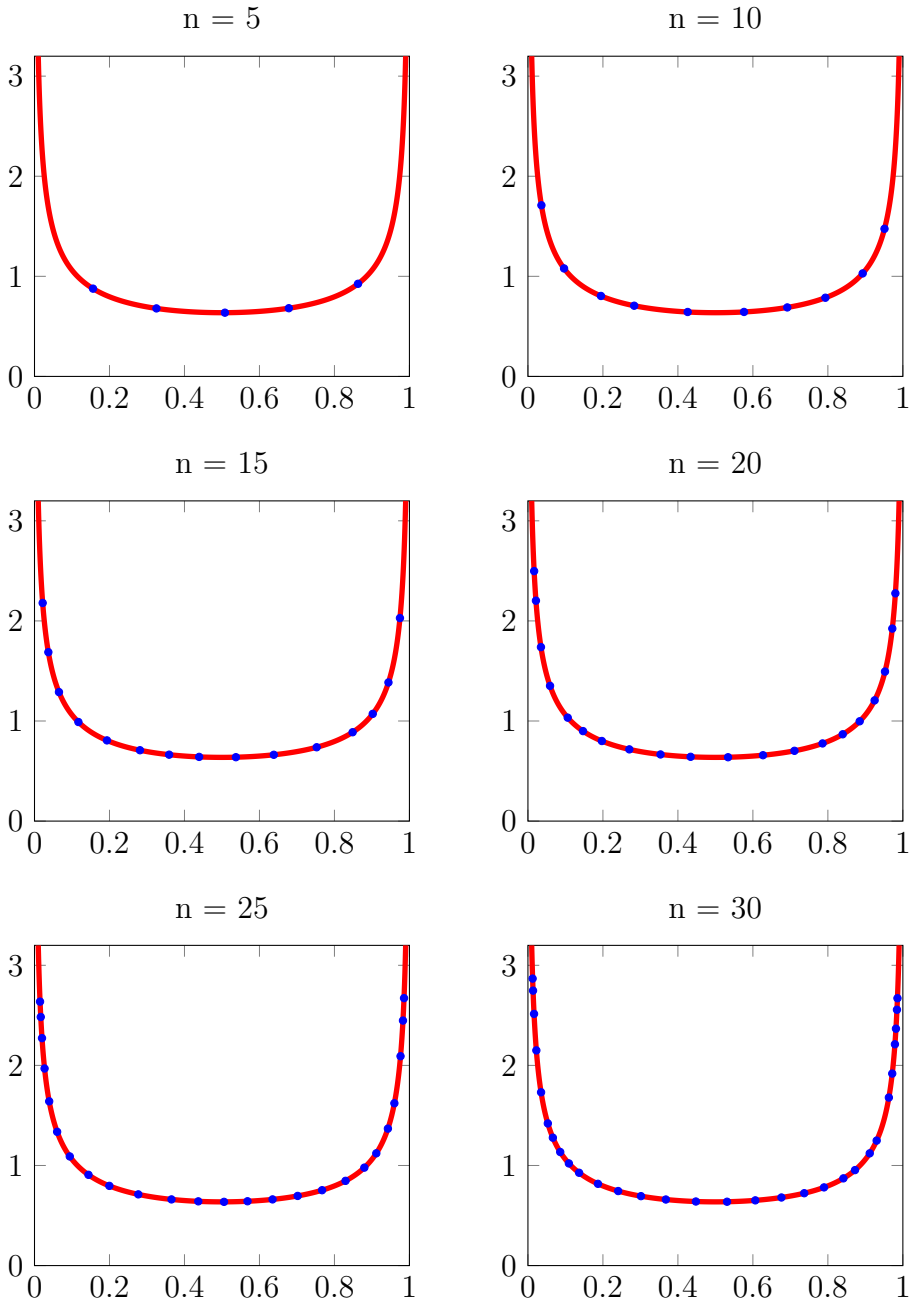

Figure 5.4: Output Space-filling for $Beta(0.5, 0.5; x)$.

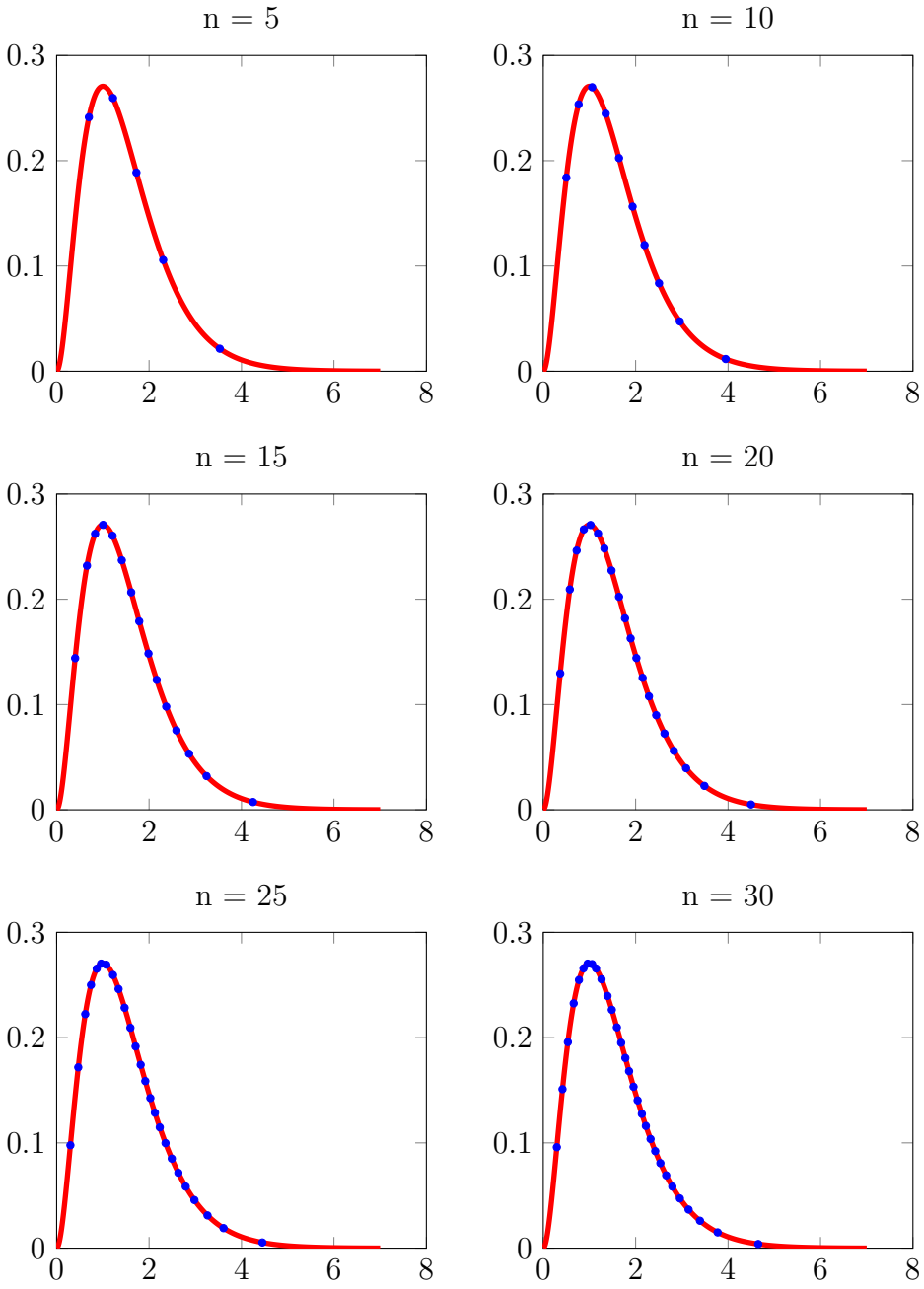

Figure 5.5: Output Space-filling for $\Gamma(x; 3, 2)$.

For f_6, we will choose Domain $= [-20, 20]$ and Bound $=[0, 0.1]$.

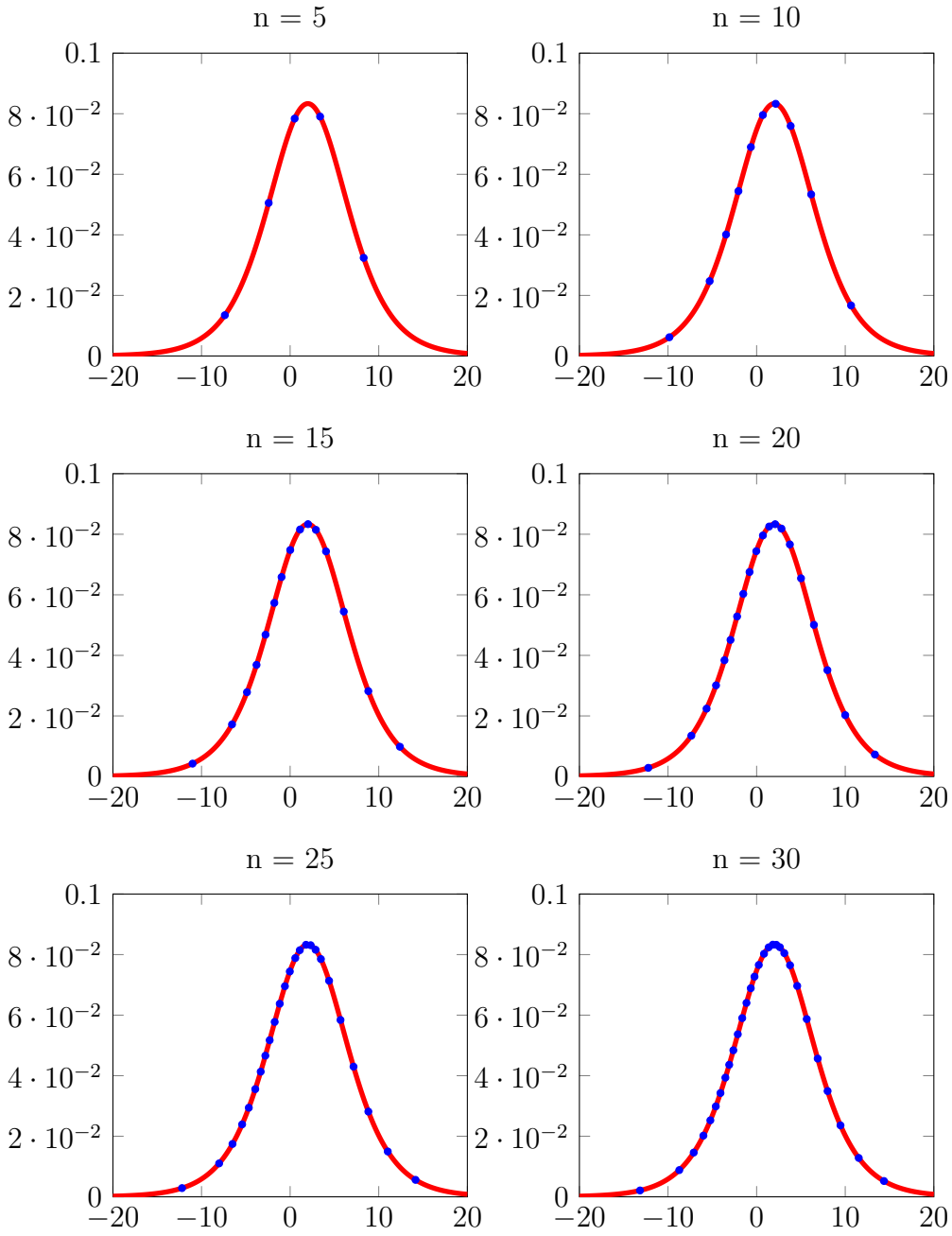

Figure 5.6: Output Space-filling for $Logit(x; 3, 2)$.

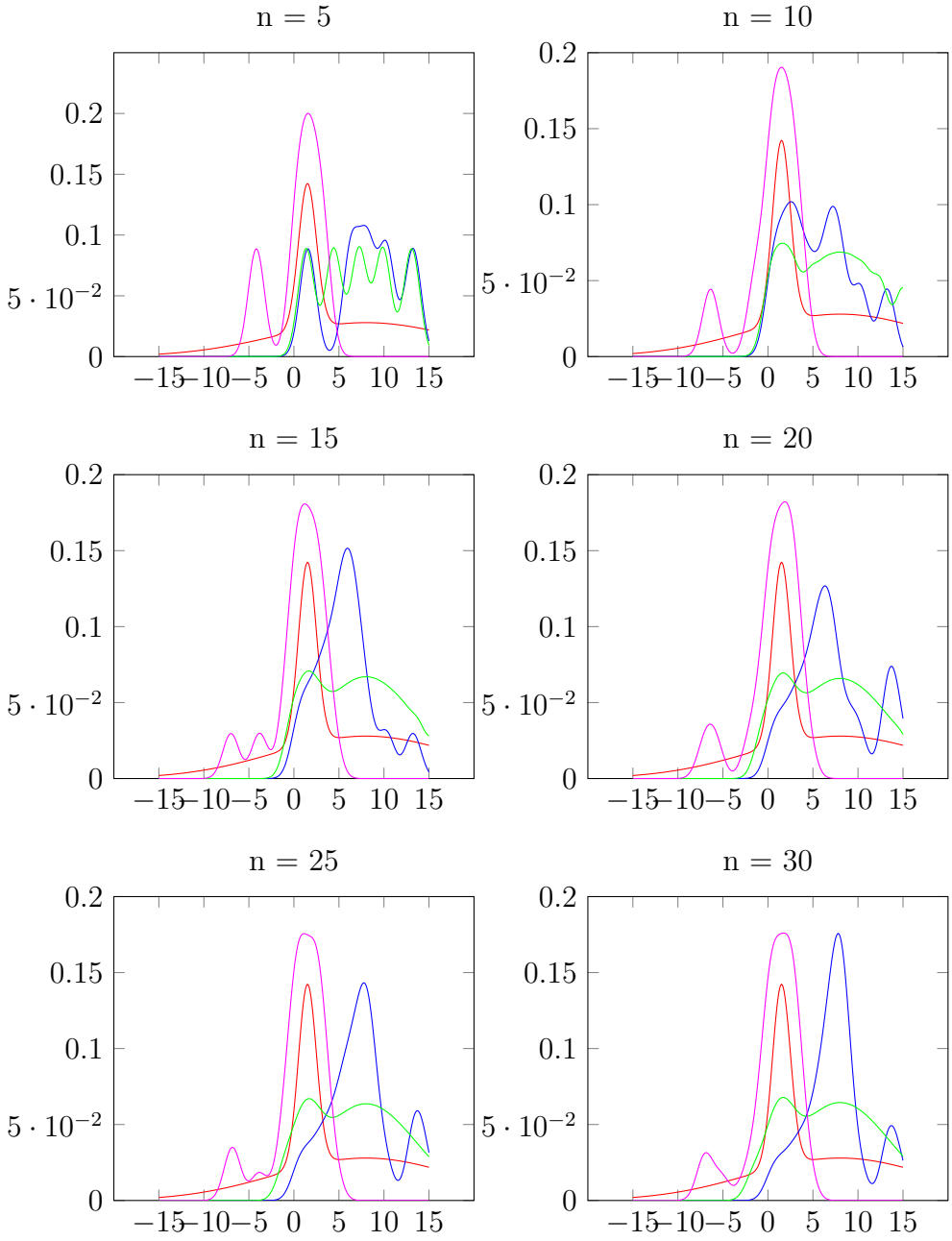

Figure 5.7: **Mixture Normal Estimation Comparison**(f_1).
Actual function(red), MERPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.9$.

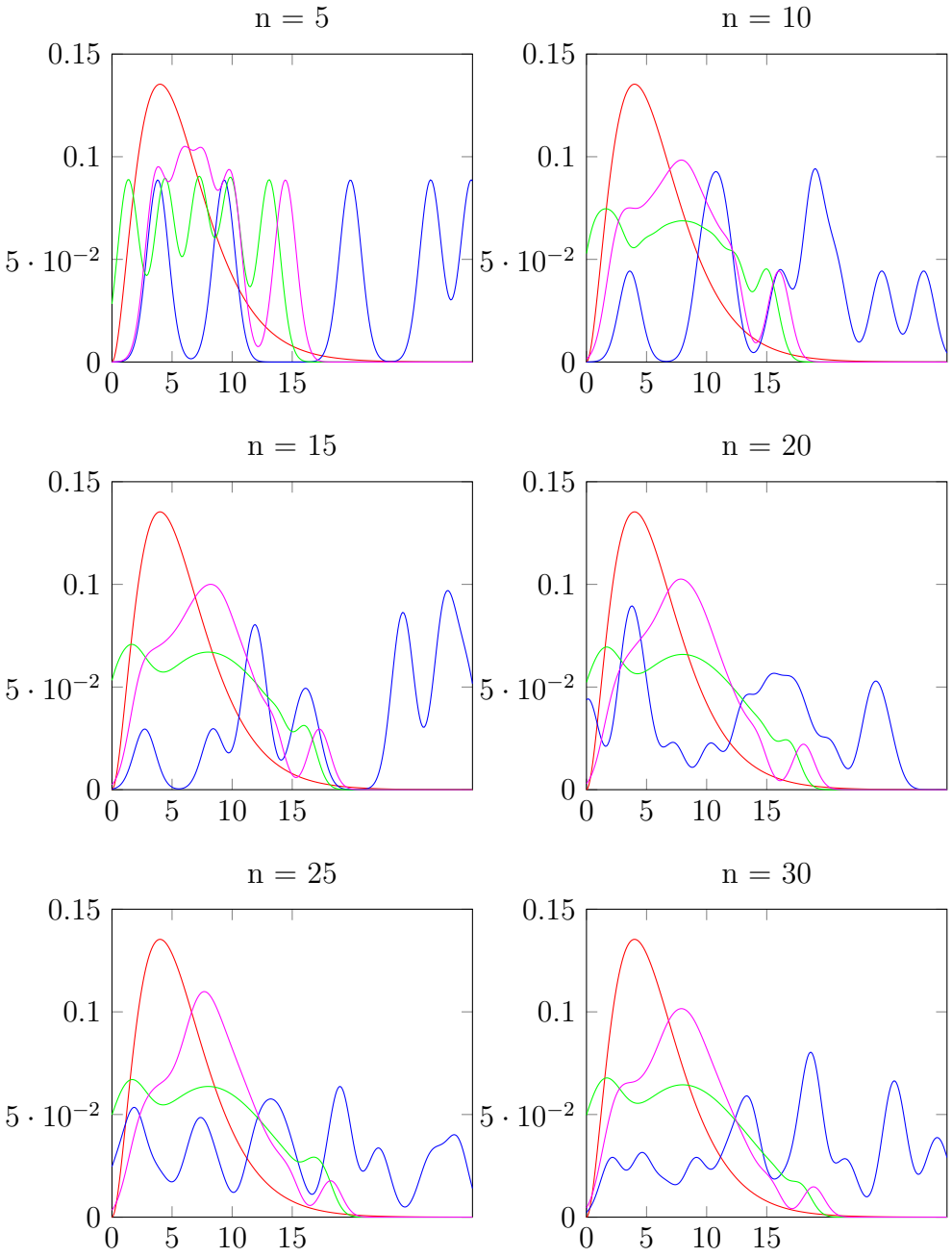

Figure 5.8: χ^2 **Estimation Comparison**(f_2).
Actual function(red), MCRPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.9$.

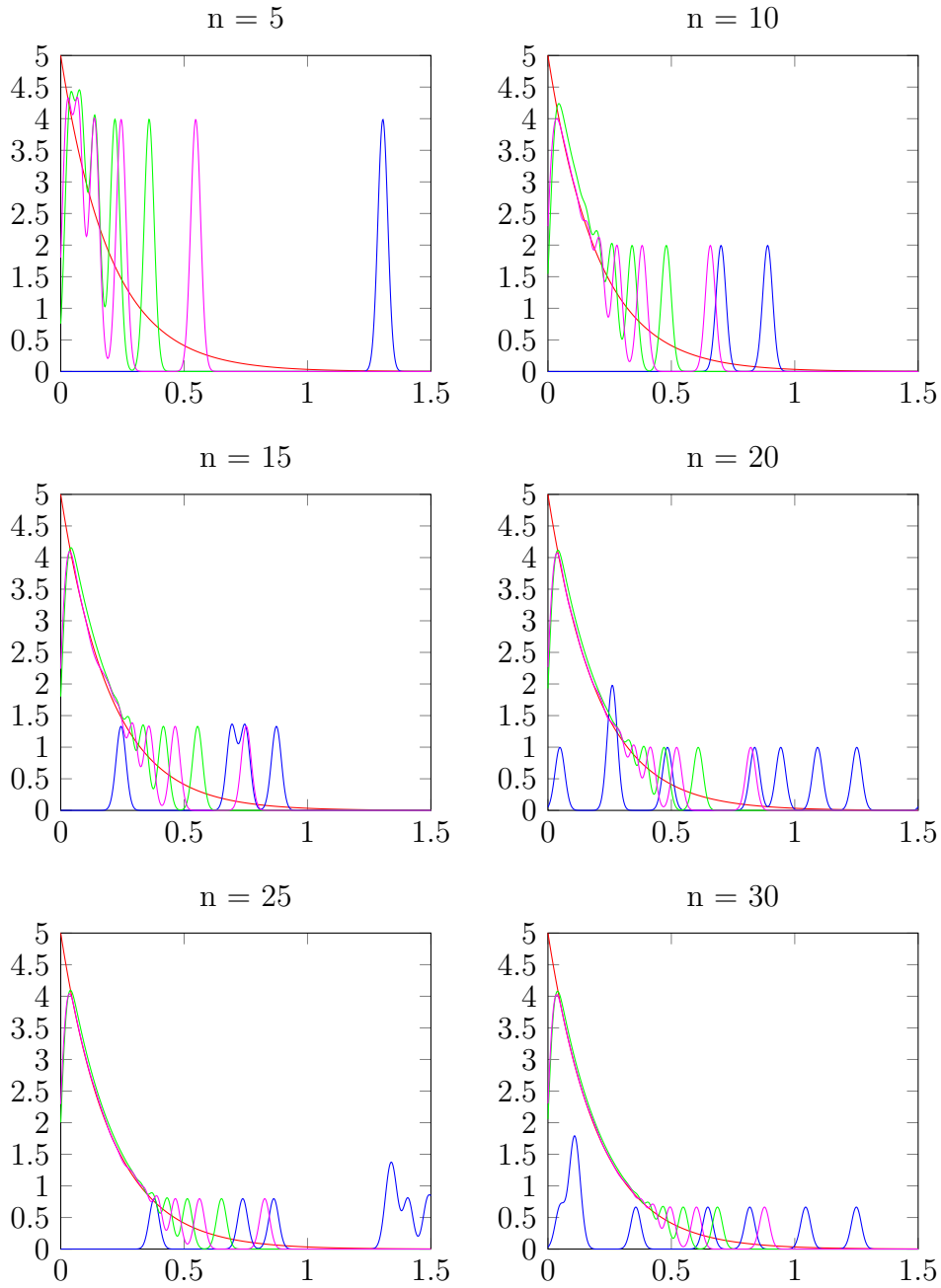

Figure 5.9: **Exponential Distribution Estimation Comparison**(f_3).
Actual function(red), MCRPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.04$.

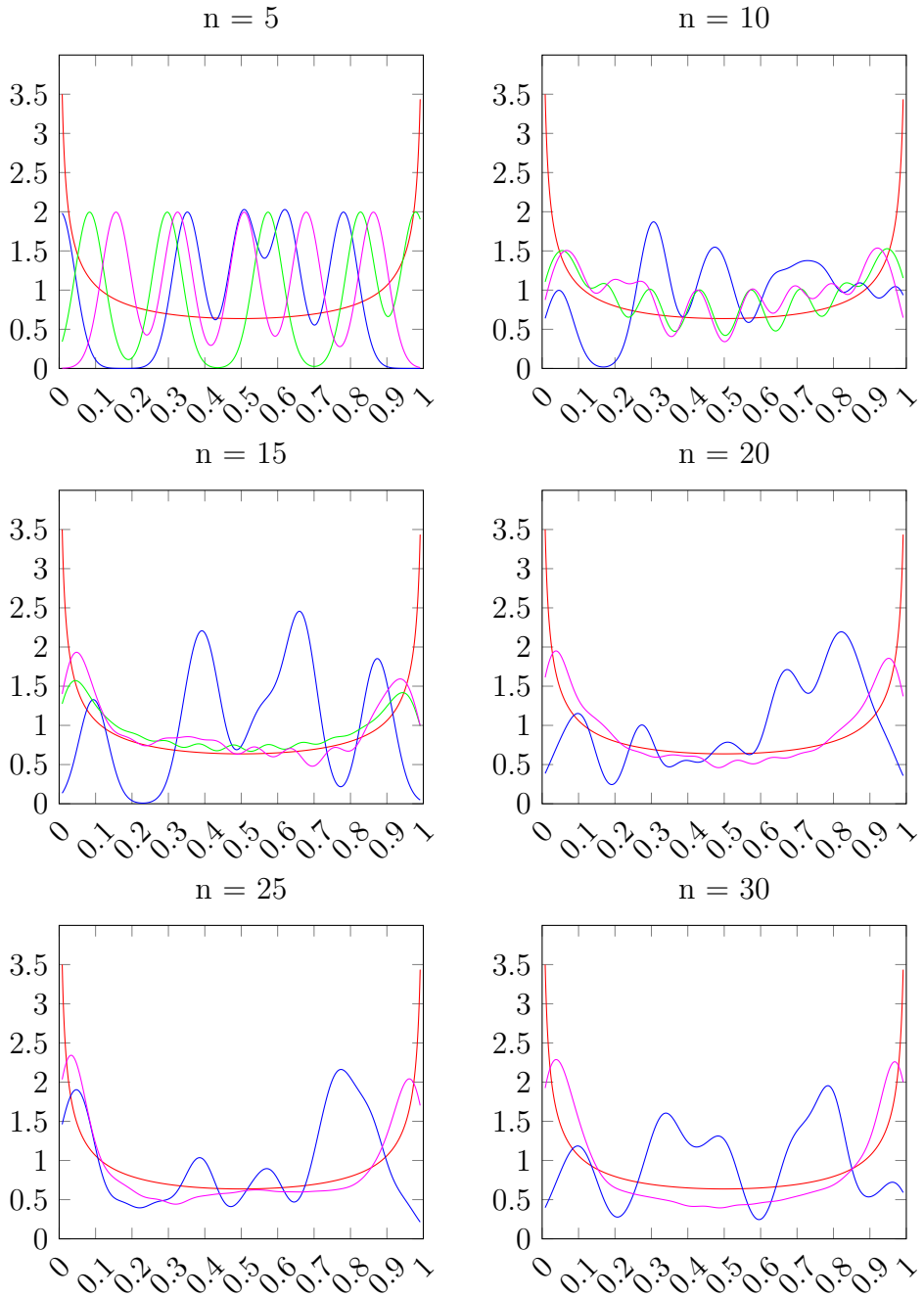

Figure 5.10: **Beta Distribution Estimation Comparison**(f_4).
Actual function(red), MCRPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.04$.

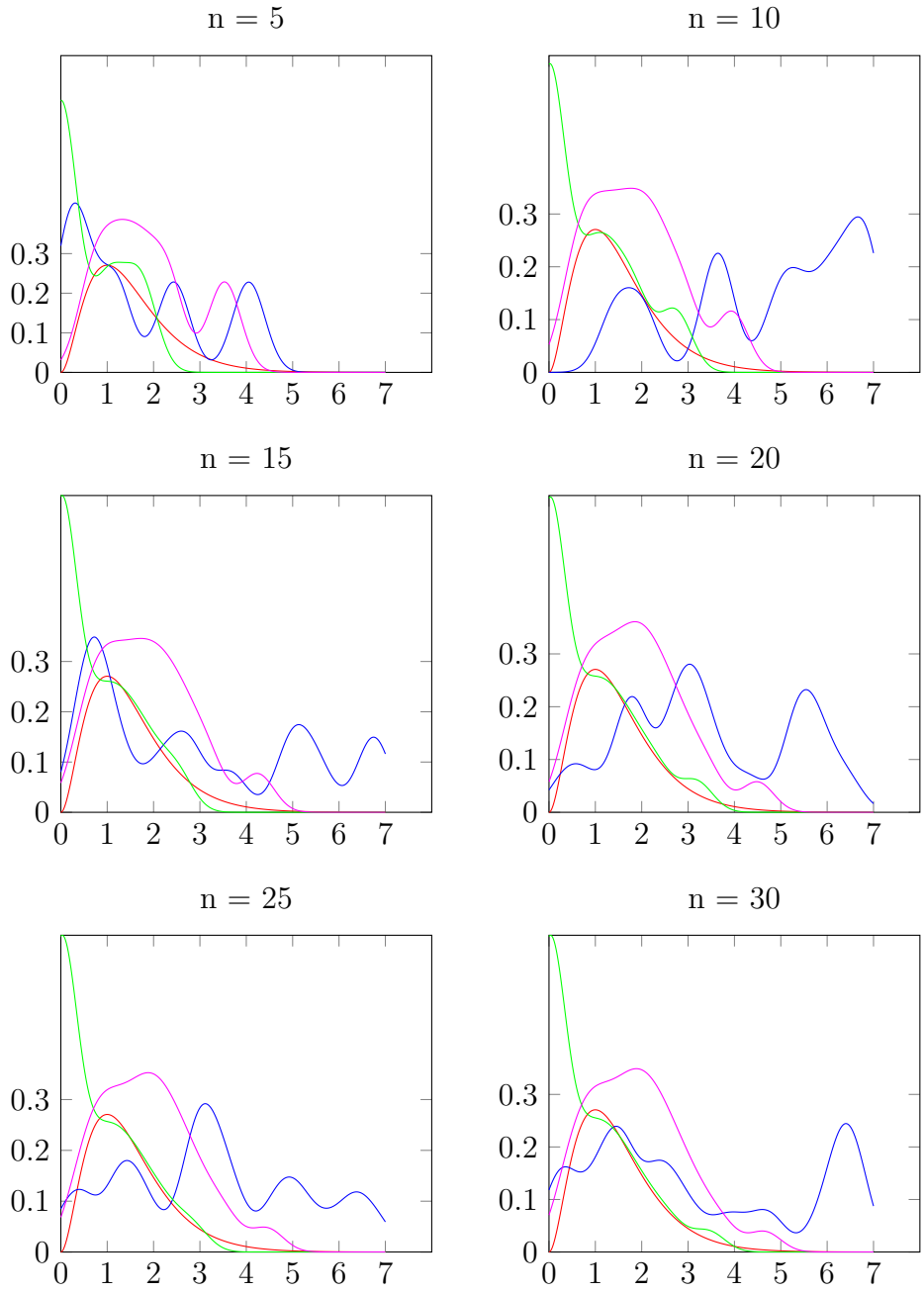

Figure 5.11: **Gamma Distribution Estimation Comparison**(f_4).
Actual function(red), MCRPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.5$.

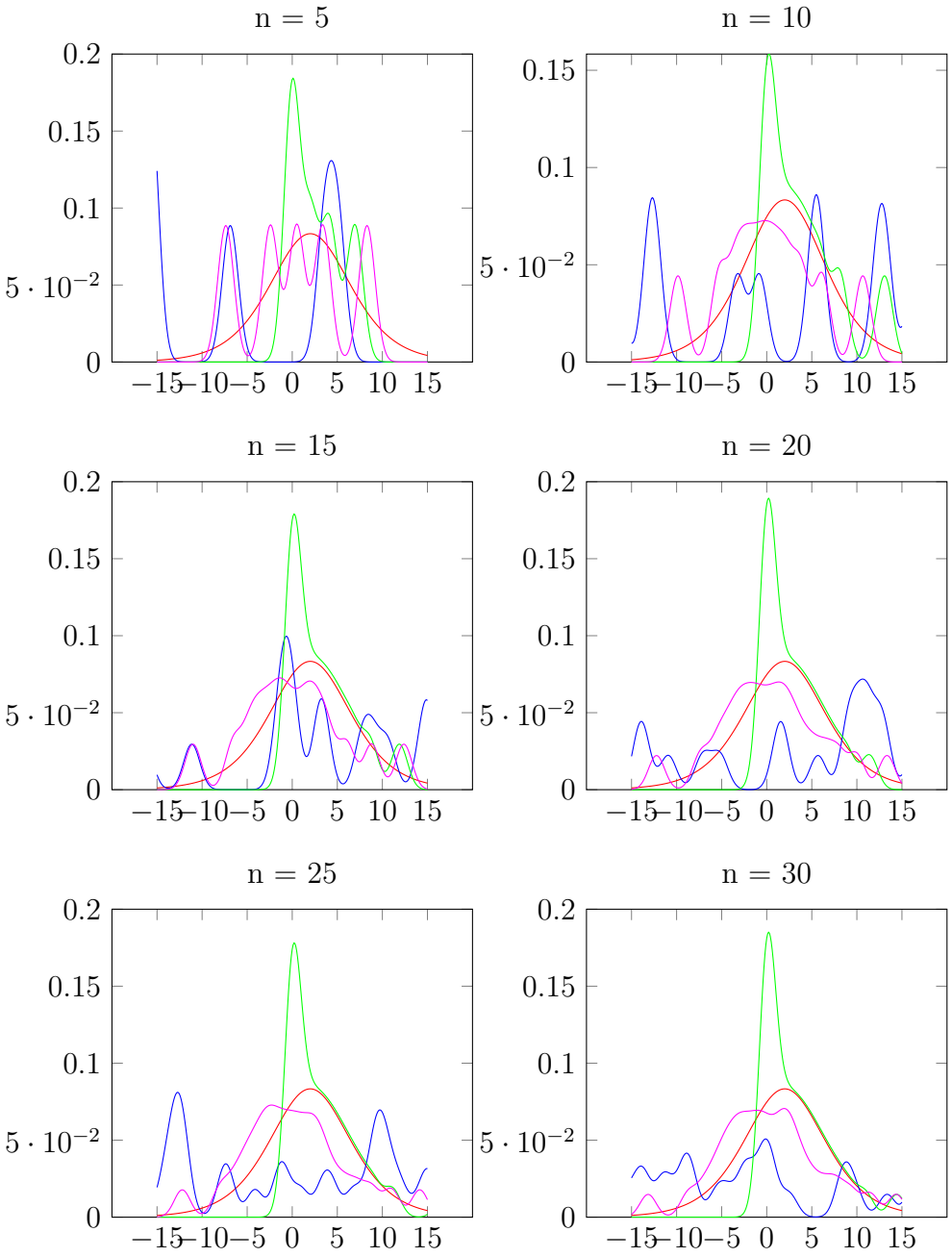

Figure 5.12: **Logistic Distribution Estimation Comparison**(f_6).
Actual function(red), MCRPs KDE(blue), MMSERPs KDE(green), OSF KDE(magenta). By sample KDE, with $h = 0.9$.

Input Sample for f_1																				
$n=1$	2.785561551																			
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...				
$n=13$	-7.169438	-0.92636542	0.15696588	0.54215957	0.9362542	1.31333189	1.737184498	2.137211	2.52583826	2.948872546	4.9294165	9.911875248	2.774767							
$n=14$	-8.16288931	-1.321192151	0.8154722	0.4332648	0.724467735	1.9641942	1.492575189	1.847449341	2.24428459	2.568231578	2.9782243	4.831467714	1.634263	19.99855256						
$n=15$	-7.987651633	-1.4916125	-0.863648	0.37414984	0.69282782	1.9112397	1.3562775	1.77319288	2.1413529	2.471587974	2.79757423	3.1779584	4.9227143	11.64374632	21.3222315					
$n=16$	-7.298547159	-1.362518333	-0.95659567	0.32912795	0.643329127	0.937551172	1.25374839	1.639568436	1.98429538	2.354234472	2.616749644	2.9243886	3.6567288	6.48352867	12.37216834	22.64139543				
$n=17$	-8.3714895	-1.97168542	-0.273681422	0.92294355	0.357515624	0.64741426	0.941414513	1.364998248	1.76469951	2.28591979	2.3253165	2.62173553	2.897328	3.43774249	5.86372937	12.29158557	21.2536717			
$n=18$	-8.199661984	-2.16784986	-0.2747671	0.161427717	0.423878515	0.7269687	1.23283651	1.35469859	1.778321	1.99146576	2.299611163	2.549128148	2.81299	3.73227619	3.456639792	6.372363498	12.92657713	22.76476369		
$n=19$	-9.43889269	-2.25361344	-0.357468138	0.6921665	0.34395	0.54379924	0.82145819	1.99884757	1.468471535	1.8169625	2.13684313	2.4877583	2.6445124	2.878811667	3.13487772	3.895286833	7.96319992	13.77482856	22.2411848	
$n=20$	-1.884481	-2.3573766	-0.42916491	0.7298877	0.324442	0.56873599	0.82491766	1.7634936	1.372317617	1.67788754	1.984248289	2.2518465	2.53655916	2.712344389	2.95791477	3.24462189	4.26533577	8.98515757	14.63533594	23.18222763
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...				
$n=21\sim30$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Input Sample for f_2																				
$n=1$	7.246068426																			
$n=2...$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...				
$n=13$	0.108429375	1.406075956	2.464611073	3.67690718	4.797712887	5.860177048	6.882578685	7.730693552	8.673622977	9.711475351	10.85383576	12.50298906	17.18768004							
$n=14$	0.169871686	1.556582281	2.597451126	3.535497789	4.574258203	5.537278284	6.485402713	7.340911046	8.143425108	9.089373561	9.928595293	11.03688467	12.68520963	17.08109806						
$n=15$	0.083437536	1.344790733	2.478797169	3.474830413	4.341149658	5.288254539	6.17086399	6.99865819	7.735387131	8.536048439	9.316605915	10.31385932	11.34468606	12.93242707	16.86949676					
$n=16$	0.056860895	1.116224971	2.256632072	3.20858623	4.134142129	5.004766452	5.926676172	6.652322394	7.377409512	8.124158682	8.855390511	9.645353944	10.59950253	11.7200257	13.23951613	17.24316094				
$n=17$	0.18074774	1.068007503	1.927915164	2.760032598	3.599706578	4.617716371	5.573688148	6.240298266	6.900754913	7.645140832	8.367624467	9.070652475	9.877276498	10.83282394	12.05270148	13.51298742	16.78174328			
$n=18$	0.203020775	0.81155604	1.742431842	2.576720515	3.434597354	4.369147828	5.07768747	5.924924512	6.647169644	7.245626416	7.881209797	8.487886304	9.139888679	9.978894487	10.84704161	12.05611409	13.58446946	17.09097193		
$n=19$	0.346195308	0.646544568	1.676009116	2.500185764	3.243082838	4.025417702	4.944095589	5.68214733	6.386547866	7.004631713	7.59527683	8.187615926	8.788956963	9.471944906	10.14983651	11.09244154	12.11683944	13.77487043	17.50399436	
$n=20$	0.475564442	0.554235269	1.374463713	2.220719177	3.117691628	3.87033284	4.732968438	5.483700423	6.22323806	6.831526589	7.361717589	7.928274541	8.456995125	8.992502789	9.646123286	10.39925938	11.21280346	12.29547322	13.81044975	18.00071019
$n=21\sim30$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Input Sample for f_3																				
$n=1$	0.431631368																			
$n=2\sim12$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
$n=13$	0.013620096	0.029764391	0.046585371	0.064200709	0.0867847	0.110830852	0.142389726	0.173888317	0.218039963	0.265477127	0.338932811	0.433761148	0.775019978							
$n=14$	0.012535998	0.028136808	0.044420612	0.058894046	0.079973479	0.101200355	0.130906165	0.156937343	0.189795054	0.228742732	0.275643553	0.3453428	0.455101098	0.766034131						
$n=15$	0.010086111	0.028183942	0.04167849	0.058031139	0.075949744	0.095368586	0.113319452	0.145130191	0.171882286	0.204963228	0.240448068	0.296341577	0.360691724	0.469341279	0.755826551					
$n=16$	0.010270239	0.022620476	0.034818048	0.051626665	0.069684203	0.087366547	0.105946087	0.130865469	0.154681334	0.178178489	0.217774223	0.262553766	0.308866451	0.377041142	0.47736129	0.75981994				
$n=17$	0.005957797	0.016174743	0.0311057	0.049640162	0.06682942	0.079542398	0.099743579	0.117661096	0.141725444	0.163728143	0.195581708	0.227925432	0.26764077	0.316660513	0.393807708	0.490684125	0.730301114			
$n=18$	0.010250038	0.018472125	0.031510072	0.045927537	0.059738263	0.076309728	0.092740046	0.107573723	0.131073067	0.152539914	0.176058888	0.20532953	0.238672188	0.282208705	0.325305367	0.399443932	0.502224313	0.777989666		
$n=19$	0.008397473	0.019218156	0.029244322	0.042569472	0.054789402	0.070089018	0.084408128	0.101424034	0.122683274	0.142601758	0.161718117	0.185905197	0.214116541	0.251346118	0.289333816	0.335033328	0.406236006	0.511582141	0.777584673	
$n=20$	0.009582411	0.019302949	0.027503199	0.039537566	0.050284385	0.064794724	0.08022531	0.09729349	0.112747674	0.13148693	0.150876728	0.170319096	0.197338867	0.226214537	0.261230125	0.297500604	0.348479863	0.416432845	0.51940036	0.824544369
$n=21\sim30$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 5.1: OSF results for $f_1 \sim f_3$ (Part of)

Input Sample for f_4																							
$n=1$	0.443113978																						
$n=2\sim12$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
$n=13$	0.011219565	0.015683577	0.026456324	0.076542513	0.151587185	0.318662892	0.493439896	0.680660661	0.833550461	0.92392586	0.971658778	0.984829085	0.989672495										
$n=14$	0.011361914	0.021562652	0.026434118	0.067389355	0.140683456	0.257397257	0.392808197	0.530175631	0.735715252	0.866477973	0.948377291	0.973392536	0.985132452	0.989068605									
$n=15$	0.011366291	0.014804369	0.02126863	0.051897083	0.099587676	0.187633044	0.331421473	0.517794091	0.684060299	0.797822204	0.88462242	0.940097947	0.972072037	0.985679482	0.989191182								
$n=16$	0.011517282	0.016056324	0.026514643	0.047858484	0.089064413	0.1382628	0.257008527	0.425751574	0.594484573	0.730340039	0.851100077	0.909893927	0.957152631	0.978308961	0.98618045	0.988863069							
$n=17$	0.011603374	0.013849094	0.019519724	0.033781559	0.062352793	0.115299839	0.220412299	0.361180497	0.526869388	0.665554894	0.78785453	0.892337608	0.941724677	0.965742599	0.977194531	0.987015231	0.989594752						
$n=18$	0.011926729	0.013114268	0.022176526	0.032683179	0.049025206	0.094578285	0.166783353	0.268729319	0.41868575	0.577216888	0.726921686	0.85545174	0.931500485	0.963373648	0.978231272	0.985231125	0.986804853	0.98991328					
$n=19$	0.011117066	0.012305052	0.016056356	0.022811434	0.038921745	0.070354206	0.124082555	0.212965004	0.3492925	0.528804213	0.675101341	0.796919801	0.868477204	0.920631524	0.955834618	0.973038647	0.983578885	0.987648692	0.990255485				
$n=20$	0.009675273	0.012317045	0.015763183	0.021654202	0.033207185	0.060429476	0.112549502	0.184100744	0.308035711	0.455088389	0.604835718	0.745070761	0.841906235	0.916951439	0.955159536	0.97275987	0.985277988	0.988454325	0.990202243				
$n=21\sim30$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...			

Input Sample for f_5																							
$n=1$	1.199472408																						
$n=2\sim12$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
$n=13$	0.268068307	0.300794327	0.31119002	0.32302795	0.332659122	0.336602417	0.831285436	1.212542863	1.617297313	1.994826594	2.366969646	2.825186992	4.009471529										
$n=14$	0.029449435	0.291867186	0.302493238	0.313069198	0.324364312	0.333981657	0.553496919	0.963339095	1.375657042	1.74176491	2.065651917	2.410782072	2.866590368	3.983511019									
$n=15$	0.129387617	0.29551821	0.305936504	0.315524462	0.324411303	0.333395878	0.393767442	0.785283867	1.152200911	1.511641816	1.844476882	2.141051735	2.430415832	2.903125259	3.946832396								
$n=16$	0.178743839	0.277501709	0.297364706	0.307533845	0.318017498	0.328210874	0.334101644	0.626917128	0.994063963	1.298560571	1.643708026	1.952903668	2.243056357	2.55199166	2.976209095	3.942611145							
$n=17$	0.084773533	0.280462897	0.301359745	0.309915945	0.319122587	0.329647866	0.334537994	0.385810218	0.793410235	1.172135168	1.488190709	1.748110123	1.974843953	2.241538573	2.598367322	3.013030993	3.88641329						
$n=18$	0.214673197	0.280212805	0.293680143	0.303812434	0.312980978	0.319312188	0.326856563	0.334581605	0.669106794	0.98955466	1.306106537	1.565889278	1.84946063	2.070430094	2.321131858	2.614639423	3.044453669	3.987299813					
$n=19$	0.057143046	0.278570137	0.297553071	0.304958356	0.313207658	0.32062591	0.327692823	0.335331199	0.46843025	0.787542422	1.096298634	1.373081467	1.64774874	1.904976267	2.118008335	2.374448055	2.678927181	3.109348588	4.021316893				
$n=20$	0.155175053	0.261758851	0.286201338	0.299745701	0.307497061	0.314307583	0.321124117	0.327595342	0.33536781	0.594629781	0.896742795	1.210678432	1.504717862	1.745231748	1.970046582	2.178746588	2.714771032	3.143002241	4.169200774				
$n=21\sim30$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Input Sample for f_6																							
$n=1$	1.601736319																						
$n=2\sim12$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
$n=13$	-9.626318476	-5.464131039	-3.531596461	-2.085146838	-0.553086273	0.913048044	1.836403919	2.809070498	4.237392028	5.676682715	7.352971083	9.189628434	13.7856148										
$n=14$	-10.33081393	-5.901016917	-3.74268189	-2.49632681	-1.160663484	0.250020502	1.372419756	2.909690048	3.301114283	4.585182773	6.117268406	7.682394347	9.471714067	13.36603857									
$n=15$	-10.24872845	-6.138353004	-4.120080131	-2.832678627	-1.48572781	-0.079085407	0.971485098	1.863387227	2.91598862	3.88821583	5.194608732	6.356987690	7.839428354	9.698348014	13.54226102								
$n=16$	-10.30132455	-6.397499209	-4.511934355	-3.120279068	-1.899768093	-0.828321436	0.148715139	1.156257182	2.165660164	3.084561709	3.993952303	5.151999258	6.29008898	7.953289373	9.99967991	13.88479932							
$n=17$	-9.852684102	-6.418184342	-4.710378696	-3.335116998	-2.265964463	-1.264710329	-0.189776024	0.708079973	1.572920105	2.428180388	3.202859016	4.30354723	5.49753879	6.784679778	8.242464512	10.20230243	13.99482922						
$n=18$	-10.55031854	-6.870952355	-5.070116802	-3.624446281	-2.487406368	-1.285725692	-0.155507716	0.586418476	1.429654846	3.130056024	3.955091639	4.921998605	5.725851623	6.925851002	8.344727406	10.13830914	14.22580575						
$n=19$	-10.90114405	-6.893661428	-5.248728105	-3.918205774	-2.813032657	-1.600295679	-0.460526366	0.563943781	1.364223955	2.056017814	2.76038456	3.553944653	4.356645353	5.38208791	6.438290911	7.475313666	8.731885643	10.54832523	14.25118417				
$n=20$	-10.82081386	-6.790209833	-5.038531866	-3.703828103	-2.78918404	-1.663740872	-0.756105866	0.236824547	1.082920848	1.782024705	2.475911365	3.107571757	3.999845558	4.858560049	5.723120629	6.746346401	7.888814977	9.105567263	10.94253755	14.89032729			
$n=21\sim30$	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 5.2: OSF results for $f_4 \sim f_6$ (Part of)

5.2 Multivariate Output Case

In this section, the property of space-filling property will roughly be evaluated based on the result of function computational method. we will use following function to attain this target

- $f_7 : \mathbb{R}^2 \to \mathbb{R}^2$, $(X, Y) = f_7[x, y] = \left(\frac{1}{\sqrt{x^2 + y^2 + 0.1}}, \arctan \frac{y}{x} \right), x \in [0, 1], y \in [0, 1]$

- $f_8 : \mathbb{R}^2 \to \mathbb{R}^2$, $(X, Y) = f_8[x, y] = \left(x^2 + y^2, xy\right), , x \in [0, 2], y \in [0, 2]$

Since we have already obtained Output space-filling sample (OSF), it is of great necessity to make evaluation on the goodness of space-filling. Thus, we will use following definitions to make evaluation on the goodness of space-filling property in output space.
The following scatter plot marked LHD (transformed to a specific range) in orange and OSF image (mapping result in range space) in blue. Intuitively, these two kinds of sampling result have attained space-filling. Then, we compare Latin Hypercube Design(LHD) with the image of OSF(LHD) under a soft predetermined boundary for following functions, where OSF(D_m^n) means that this OSF is filtered by a design with m rows, n factors. Now, we will apply Cosine Similarity to verify it. The cosine similarity(denoted as $S(a, b)$) is defined by the formula of dot product. Given two vector a and b, the cosine similarity defined by

$$S(a, b) = \frac{||a \cdot b||}{||a|| \cdot ||b||} = \frac{\sum_{i=1}^{n} a_i \times b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}} \tag{5.9}$$

When the value of $S(a, b)$ approximate to 1, which means the point a and b are similar to each other. On the contrary, it means that the a and b are far away.
We apply this technique on f_7, then we obtain

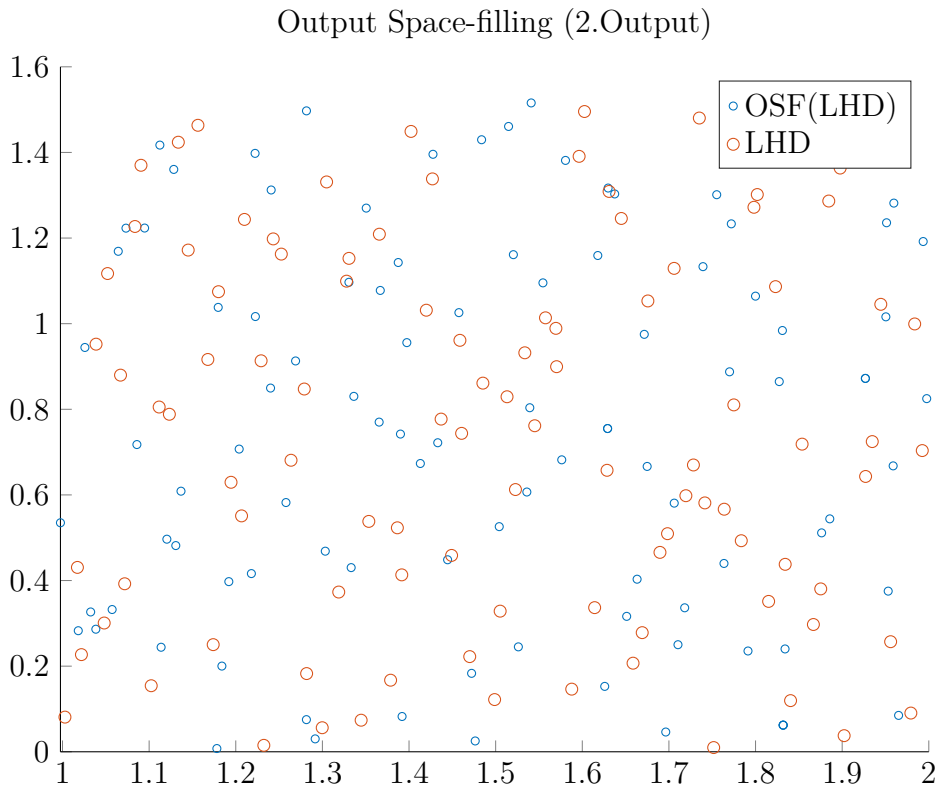

Figure 5.13: Space-filling in range space for f_7, $n = 100$

Function	Label	min	max	mean	median
f_7	OSF(LHD) v.s. LHD	0.9939	1.000	0.9993	0.9997
f_8	OSF(LHD) v.s. LHD	0.9937	1.000	0.9994	0.9997

Table 5.3: Cosine Similarity of f_7 and f_8

5.3 Conclusion and Future Work

In uni-variate output situations, the OSF shows excellent property in kernel density estimation. And it also passed the Cosine Similarity with higher than 96% similarity in the OSF test for two output variables. In this process, we have already found that in the uni-variate output space-filling, it can be thought as an excellent substitution of RP in lot of cases. The main advantage of it is solving the out of bound problem of generating for Beta distribution via predefined constrain.
In this thesis, we have already introduced the generation of Bounded Output Space-filling methodology called NFF. And we provide six uni-variate output cases, which are probability density functions to generate the input points which can be used as the RPs and obtain a good result in the kernel density estimation. We also applied the two sample Cosine Similarity to output with a LHD filtered and the original LHD. We found that the LHD filtered image of OSF is also be uniform as we expected. In the future, there are lots of possible researches can be performed.

- Update this NFF methodology for higher running speed and accuracy. The NFF method contains a process of re-selecting the candidates iteration process. It is really time and computational consuming.

- Apply current NFF methodology to the field of marginal detection. The concepts of OISF and OBSF in this thesis have been proposed, but they are not been detailed discussed. From the high adaptive property of the closet point iteration, it can also be applied to non-convex circumstances.

- When generating OSF for Beta function in Uni-variate output case. The output value can be out of the bound for ordinal RP generations. Thus, it can be thought as an enhanced RPs and have more works on it. And it is also possible to generate the weight of each point from the perspective of as RP, and it can be thought as a discrete population in kernel density estimation.

- For NFF methodology can be used in the training of a machine learning model. Since it is applicable for multivariate input and output functions.

Appendix

Appendix A: BOSF MATLAB code

Code for generating output boundary space-filling sample. (matlab r2021a)

```
function Invsf = losfdesign(f,g,Domain,N, D)
% This function can be used to generate the output space-
   filling sample, and the detailed parameter
   illustrations are as follows
%% f: an input function handle which parameters should be
   in separate variables form.
%  eg. f = @(x,y) [1./sqrt(x.^2+y.^2+0.1),atan(y./x)];
%% g: an input function handle which parameters should be
   in matrix form.
%  eg. g = @(X) [1./sqrt(X(:,1).^2+X(:,2).^2+0.1),atan(X
   (:,2)./X(:,1))];
% Note:
% 1. Function f and g are same, the only difference of
   them is the format.
% 2. Order of variables should be same, like x is the
   first variable in function handle f, and it also should
    be the first column of matrix X in function g.
%% Domain: an input matrix which contains domains of all
   the variables.
% eg.Domain = [LD1, UD1;
```

```
  %               LD2, UD2;
  %               ..., ...;
  %               LDm, UDm]; for m input variable case.
  %    LDi: Lower bound of domain for i^th variable.
  % Note: In each row, the LB should be less than UB, and
     the input order cannot be reversed.
  %% N: the number of runs in the output space-filling
     sample will be generated.
  %% Invsf: result for output space-filling sample.
  %% Design: The pre-defined the design in the input, which
     will be used as filter to create the space-filling case
     .
  %Note: After generation, the coefficients of previous
     function can be substituted by the result, and the
     output will attain the target of output space-filling.
[a,~]= size(Domain);
Invsf=[];

[MU] = maxfind(f,g,Domain);
aa = ndim_Normal(10^a*N,MU);
aa = filter(aa, Domain);
Invsf = Min_dis_filter(aa,g(aa),Bound,N,Design);%
   Min_dis_filter(Invsf,g(Invsf),Bound,N,Design);
end


function [M] = maxfind(f,g,Domain)
st = func2str(f);
if  contains(st,']') == 0
M = (max(Domain)-min(Domain))/2;
c = ",";
```

```
a = strsplit(st,' ');
Out = [];MAX = [];MIN=[];idx = [];
[m,~] = size(Domain);
else
a = strsplit(st,"[");
b = strsplit(a{2},"]");
c = strsplit(b{1},",");
Out = [];MAX = [];MIN=[];idx = [];
[m,~] = size(Domain);
end
for i=1:length(c)
fi = strcat(a{1},c{i});
fi = str2func(fi);
for j = 1:m
Precise = Domain(j,1):0.01:Domain(j,2);
X = (Domain(j,2)-Domain(j,1))*lhsdesign(length(Precise),m)+
   Domain(j,1); % Use LHD is only it is easily generated by
   MATLAB.
Out = g(X);
if sum(f(Precise) > 0) == m
[~,x] = findpeaks(f(Precise));
elseif sum(f(Precise) < 0) == m
[~,y] = findpeaks(f(Precise));
else
[~,x] = findpeaks(f(Precise));
[~,y] = findpeaks(-f(Precise));
end
idx = [idx,x];
idx = [idx,y];
MAX = [MAX;X(x,:)];
MIN = [MIN;X(y,:)];
```

```
end
M = X(idx,:);
end
end


function [Input] = ndim_Normal(k,mu)
Input = [];
[a,b]= size(mu);

for j=1:a
Covmat = 10^b*diag(ones(1,b));
i = 0;
core=[];
while i < k
r = randn(1,b)*chol(Covmat)+repmat(mu(j,:),1,1);
core = [core;r];
i = i + 1;
end
Input = [Input;core];
end
end


function [Out] = filter(Input,Domain)
[a,b] = size(Input);
for i = 1:a
for j = 1:b
if (Input(i,j)-Domain(j,1))*(Domain(j,2)-Input(i,j)) < 0
Input(i,:) = ones(1,b)*Inf;
end
```

```
end
end
Input1=isinf(Input);
[inf_r,~]=find(Input1==1);
Input(inf_r,:)=[];
Out = Input;
end


function [Inv] = Min_dis_filter(D1,f_D1,Bound,n,Design)
% D1 is the sampling matrix input
% f\_D1 is the sampling matrix output
% Bound is the output space-filling interval, which should
   be written as
% Bound=[LB1, UB1;
%        LB2, UB2;
%        ..., ...;
%        LBs, UBs];

% Design is the space-filling design used as a filter.
% n is the number of runs in the final result of the output
   design
Inv = [];
LHD = Design;
[n,s] = size(Design);
for i=1:s
LHD(:,i) = LHD(:,i)*(Bound(i,2)-Bound(i,1))+Bound(i,1);
end
a = dsearchn(f_D1,LHD);
for j=1:length(a)
Inv = [Inv; D1(a(j),:)];
```

```
end
end
```

Appendix B: Complete data in table 6.2 and 6.3

Input Sample for f_1
$n = 1$	2.785561551
$n = 2$	-0.78529916	6.53453738
$n = 3$	-1.536953792	1.657645785	8.214767724
$n = 4$	-2.166987933	1.3584878	2.1534851	9.3536242
$n = 5$	-3.556754177	0.893371785	1.78227724	2.68126773	12.347397
$n = 6$	-4.6568237	0.562315986	1.3193227	2.1164177	3.623561832	12.8621569
$n = 7$	-4.9697893	0.383395477	1.7179751	1.8224228	2.46538728	5.136357616	15.9455996
$n = 8$	-6.42537425	0.613818	0.765418798	1.44541846	2.42118685	2.619682959	6.5249294	13.5921346
$n = 9$	-4.557713278	0.97655437	0.714769354	1.283674449	1.86839315	2.413969325	3.1146451	6.96915495	18.25163473
$n = 10$	-5.657564927	-0.131266821	0.4794831	1.49987684	1.6738759	2.1288575	2.615959925	3.189792235	6.94625848	18.4943519
$n = 11$	-7.16288389	-0.323352775	0.373673293	0.83626587	1.33493478	1.7394525	2.223617371	2.63961541	3.515549	7.4396412	18.92757655
$n = 12$	-7.17355985	-0.77329966	0.181826992	0.565256825	0.98814341	1.4256869	1.989936	2.395978262	2.751484665	3.854586778	8.1866762	17.7669944
$n = 13$	-7.169438	-0.92636542	0.15696588	0.54215957	0.9362542	1.31333189	1.737184498	2.137211	2.52583826	2.948872546	4.9294165	9.911875248	2.774767
$n = 14$	-8.16288931	-1.321192151	0.8154722	0.4332648	0.724467735	1.9641942	1.492575189	1.847449341	2.24428459	2.568231578	2.9782243	4.831467714	1.634263	19.99855256
$n = 15$	-7.987651633	-1.4916125	-0.863648	0.37414984	0.69282782	1.9112397	1.3562775	1.77319288	2.1413529	2.471587974	2.79757423	3.1779584	4.9227143	11.64374632	21.3222315
$n = 16$	-7.298547159	-1.362518333	-0.95659567	0.32912795	0.643329127	0.937551172	1.25374839	1.639568436	1.98429538	2.354234472	2.616749644	2.9243886	3.6567288	6.48352867	12.37216834	22.64139543
$n = 17$	-8.3714895	-1.97168542	-0.273681422	0.92294355	0.357515624	0.64741426	0.941414513	1.364998248	1.76469951	2.28591979	2.3253165	2.62173553	2.897328	3.43774249	5.86372937	12.29158557	21.2536717
$n = 18$	-8.199661984	-2.16784986	-0.2747671	0.161427717	0.423878515	0.7269687	1.23283651	1.35469859	1.778321	1.99146576	2.299611163	2.549128148	2.81299	3.73227619	3.456639792	6.372363498	12.92657713	22.76476369
$n = 19$	-9.43889269	-2.25361344	-0.357468138	0.6921665	0.34395	0.54379924	0.82145819	1.99884757	1.468471535	1.8169625	2.13684313	2.4877583	2.6445124	2.878811667	3.13487772	3.895286833	7.96319992	13.77482856	22.2411848
$n = 20$	-1.884481	-2.3573766	-0.42916491	0.7298877	0.324442	0.56873599	0.82491766	1.7634936	1.372317617	1.67788754	1.984248289	2.2518465	2.53655916	2.712344389	2.95791477	3.24462189	4.26533577	8.98515757	14.63533594	23.18222763
$n = 21$	-9.74718213	-2.662822	-0.652462465	-0.6497967	0.1768796	0.44459792	0.6193627	0.8317951	1.12638314	1.429846833	1.745629533	2.5472887	2.338578185	2.549139394	2.7482624	2.9668359	3.246969288	4.525843274	8.62532117	14.597224	24.7824319
$n = 22$	-1.37755571	-3.2731181	-0.76628113	-0.9281118	0.122787832	0.33251629	0.567485	0.8154586	1.75141	1.313477936	1.5858872	1.86773248	2.146894447	2.367752183	2.59157726	2.799253796	3.26686396	3.5652635	4.998338266	9.4273211	15.7698881	23.175375
$n = 23$	-8.9332452	-2.6113943	-0.56934132	-0.31749197	0.25838412	0.417291	0.6994978	0.88291812	1.13878282	1.42652957	1.66548962	1.93785133	2.1626649	2.3913212	2.58462943	2.762591633	2.95146631	3.1773629	3.982961826	6.249616853	11.4781423	16.38292544	24.67668839
$n = 24$	-9.97885877	-4.62728981	-1.1436769	-0.15688643	0.111259475	0.318517753	0.5192241	0.715233849	0.94754217	1.1492378	1.41484194	1.677683683	1.921423637	2.163325626	2.41375773	2.61946864	2.7674325	2.98382319	3.198277883	4.364654	5.736971912	9.91849652	15.67441262	23.7268625
$n = 25$	-9.949862383	-3.27171989	-0.95849738	-0.2195396	0.52431496	0.238241977	0.41354257	0.612961657	0.78496937	0.98277794	1.217717623	1.484974174	1.751154515	1.987421794	2.165831799	2.365819227	2.63589831	2.779479677	2.9662553	3.219319747	4.32285687	6.733399595	11.28792218	16.737411	24.936882
$n = 26$	-1.7346992	-3.6747187	-0.99146421	-0.22181262	0.5512698	0.23228379	0.422499472	0.63172162	0.847675	1.41531289	1.287755226	1.49788273	1.76128681	1.983984277	2.191689116	2.411912164	2.557365474	2.721197183	2.89147555	3.8294229	3.382388249	4.36788781	7.1274742	12.197412	17.345266	25.3667543
$n = 27$	-12.13838347	-4.41633	-1.313946153	-0.2895842	-0.1422426	0.175562992	0.34468968	0.5986666	0.685265883	0.97567275	1.89372799	1.31314286	1.55484538	1.799259441	2.5452718	2.236189367	2.412384786	2.578344485	2.748585887	2.916274284	3.15884	3.4848795	4.2137671	7.11382199	11.9811719	17.728954	24.278391
$n = 28$	-11.137288	-3.826264958	-1.33264411	-0.298374426	-0.166422	0.145695841	0.31681567	0.4887668	0.637985	0.8137721	1.23929915	1.253554348	1.467987	1.68436158	1.895586454	2.15189447	2.3639399	2.47864333	2.6235641	2.79943635	2.963594565	3.18834743	3.7393489	5.14945162	8.3167625	13.13498738	18.7869826	25.45892
$n = 29$	-1.89443546	-4.771648163	-1.658923294	-0.46375284	-0.11588444	0.85759373	0.2359978	0.3663821	0.498319918	0.6424487	0.814888721	0.97727694	1.22316678	1.51712283	1.77493979	2.26278294	2.225262967	2.378351219	2.52395946	2.675613888	2.838787422	2.9959268	3.191832416	3.723175812	5.46889596	8.53156535	12.974738	17.6852994	24.8769398
$n = 30$	-11.5819895	-4.824326969	-1.74231728	-0.5945172	-0.16439537	0.31635258	0.1937158	0.35473737	0.49142615	0.6463763	0.782571638	0.97558481	1.1921424	1.41187732	1.627667173	1.882864495	2.77495835	2.243949754	2.414799763	2.578627766	2.731618439	2.898943894	3.68488646	3.2956675	3.91913235	5.57522674	9.42711714	13.42654689	18.15418933	25.5294768
Input Sample for f_2
$n = 1$	7.246068426
$n = 2$	3.957270319	10.97471993
$n = 3$	2.736266868	6.284306693	11.57529813
$n = 4$	1.991805089	5.098420576	7.928751458	12.51114377
$n = 5$	1.663073963	4.148714127	6.727252785	9.065712727	13.91330449
$n = 6$	1.354159402	3.633090573	5.733442504	7.623035594	9.893144944	14.13141751
$n = 7$	0.894519265	3.082761626	5.110767694	6.810278852	8.529612349	10.62280989	15.2073379
$n = 8$	0.879923323	2.807341181	4.478268458	6.131885863	7.676357139	9.169689256	10.87802051	15.58770767
$n = 9$	0.894552689	2.643818647	3.977794411	5.519163597	6.89113499	8.262693653	9.548880086	11.3664043	15.36329795
$n = 10$	0.641465534	2.150839761	3.66108822	5.138936727	6.349280009	7.472848082	8.568055154	9.92496129	11.77892724	15.65270549
$n = 11$	0.311809145	1.762222409	3.117932553	4.46157519	5.769473038	6.943257582	7.900968249	9.008274811	10.33762715	12.04383867	16.20368723
$n = 12$	0.516023071	1.744761279	3.131722431	4.55065845	5.523647387	6.470502433	7.418718551	8.393538802	9.470800974	10.57740929	12.16493477	15.82300015
$n = 13$	0.108429375	1.406075956	2.464611073	3.67690718	4.797712887	5.860177048	6.882578685	7.730693552	8.673622977	9.711475351	10.85383576	12.50298906	17.18768004
$n = 14$	0.169871686	1.556582281	2.597451126	3.535497789	4.574258203	5.537278284	6.485402713	7.340911046	8.143425108	9.089373561	9.928595293	11.03688467	12.68520963	17.08109806
$n = 15$	0.083437536	1.344790733	2.478797169	3.474830413	4.341149658	5.288254539	6.17086399	6.99865819	7.735387131	8.536048439	9.316605915	10.31385932	11.34468606	12.93242707	16.86949676
$n = 16$	0.056860895	1.116224971	2.256632072	3.20858623	4.134142129	5.004766452	5.926676172	6.652322394	7.377409512	8.124158682	8.855390511	9.645353944	10.59950253	11.7200257	13.23951613	17.24316094
$n = 17$	0.18074774	1.068007503	1.927915164	2.760032598	3.599706578	4.617716371	5.573688148	6.240298266	6.900754913	7.645140832	8.367624467	9.070652475	9.877276498	10.83282394	12.05270148	13.51298742	16.78174328
$n = 18$	0.203020775	0.81155604	1.742431842	2.576720515	3.434597354	4.369147828	5.07768747	5.924924512	6.647169644	7.245626416	7.881209797	8.487886304	9.139888679	9.978894487	10.84704161	12.05611409	13.58446946	17.09097193
$n = 19$	0.346195308	0.646544568	1.676009116	2.500185764	3.243082838	4.025417702	4.944095589	5.68214733	6.386547866	7.004631713	7.59527683	8.187615926	8.788956963	9.471944906	10.14983651	11.09244154	12.11683944	13.77487043	17.50399436
$n = 20$	0.475564442	0.554235269	1.374463713	2.220719177	3.117691628	3.87033284	4.732968438	5.483700423	6.22323806	6.831526589	7.361717589	7.928274541	8.456995125	8.992502789	9.646123286	10.39925938	11.21280346	12.29547322	13.81044975	18.00071019
$n = 21$	0.474055724	0.558637347	1.553179673	2.22421332	2.950912807	3.637640379	4.475662477	5.206428802	5.945950387	6.555009349	7.131178574	7.668522598	8.153822775	8.73608682	9.315830083	9.971192979	10.67034781	11.48732767	12.60304011	14.1206263	18.15799336
$n = 22$	0.355140791	0.56748956	1.38899866	2.129930318	2.909017252	3.633213966	4.356445025	4.973278564	5.616685811	6.213258911	6.798001474	7.35300909	7.829830294	8.368396815	8.855801497	9.411643055	9.988193657	10.68184543	11.59756805	12.60992051	14.16347233	18.05614647
$n = 23$	0.388815014	0.437247813	1.364516138	2.042880916	2.702739001	3.373367233	4.110586462	4.78493458	5.392620351	6.059365165	6.509511515	7.037171804	7.49385129	7.991806438	8.477530192	9.033671003	9.537825503	10.14613117	10.77964611	11.53946185	12.69659373	14.24352665	17.45010965
$n = 24$	0.327260894	0.504926275	1.242099267	1.849729791	2.521450834	3.032079156	3.672226544	4.408893136	5.037420492	5.660616257	6.205022194	6.771708707	7.247006002	7.707099912	8.20011243	8.652859087	9.160568599	9.685507588	10.31332349	11.09372349	11.87534162	12.99866068	14.47483694	17.77997329
$n = 25$	0.264846016	0.687340235	1.048896423	1.764896953	2.401248157	2.994862497	3.638551086	4.273499516	4.904559914	5.507057191	6.025692955	6.545143771	6.968281489	7.439883139	7.879597141	8.315997814	8.775925494	9.236624307	9.765988839	10.3597156	11.05405568	11.84844101	12.93479613	14.36179068	17.89653993
$n = 26$	0.272268263	0.654668651	1.252483741	1.860156201	2.465916205	3.016346673	3.62647177	4.34502267	4.971680596	5.515210328	6.019527172	6.498694427	6.959002217	7.380638797	7.829086562	8.200364546	8.681731328	9.119203141	9.592983185	10.10477208	10.66544101	11.31909513	12.15371797	13.14967416	14.65319988	18.18595884
$n = 27$	0.195621317	0.636451953	0.962970504	1.698054946	2.250306286	2.81431632	3.332599738	3.902143721	4.495217033	5.151758964	5.69518767	6.20405844	6.672340315	7.109789657	7.507111995	7.921270935	8.351979321	8.733112144	9.191557577	9.680059662	10.24007604	10.8206426	11.5370047	12.31863738	13.32753011	14.90847073	18.47183896
$n = 28$	0.218101841	0.673067614	0.949004428	1.577711255	2.075589925	2.642338777	3.183742187	3.693287986	4.287223175	4.963743094	5.541207762	6.037353511	6.56469781	6.983186523	7.395849171	7.762447303	8.146421591	8.553897805	9.001144083	9.406103684	9.861583743	10.37672819	10.97142312	11.61331589	12.39657081	13.38628008	14.892981	18.59243917
$n = 29$	0.073871847	0.700766689	0.717678316	1.405212235	1.865623444	2.403640682	2.920748269	3.512071083	4.126870751	4.697117511	5.219468217	5.716970572	6.138694904	6.634604291	7.030516013	7.423792179	7.795920315	8.189054492	8.591777422	9.007682168	9.41843321	9.854349544	10.37826104	10.95112672	11.60293025	12.40960435	13.4862648	14.85967316	18.08492754
$n = 30$	0.085494626	0.698233628	0.839128591	1.275748431	1.82848655	2.309834262	2.775548368	3.318911781	3.928305425	4.489601593	5.064180936	5.556254271	6.064711066	6.515178067	6.885130739	7.241590475	7.59083795	7.992784681	8.379291838	8.776386365	9.183068718	9.602782319	10.06071442	10.56158565	11.14655854	11.81888611	12.61513195	13.57165171	14.98948412	18.76446928
Input Sample for f_3
$n = 1$	0.431631368
$n = 2$	0.157349374	0.474021478
$n = 3$	0.099369172	0.175474193	0.460995213
$n = 4$	0.056638623	0.12296159	0.233201108	0.481356986
$n = 5$	0.038906817	0.09386611	0.156351198	0.266493777	0.579851265
$n = 6$	0.026307267	0.076037057	0.12464097	0.191120139	0.293900334	0.584041505
$n = 7$	0.038451603	0.066623599	0.101645764	0.145203783	0.220047167	0.322023425	0.630795334
$n = 8$	0.013768494	0.04608667	0.082551897	0.124936282	0.17731822	0.248768472	0.343692077	0.677949886
$n = 9$	0.019617696	0.038755081	0.07127701	0.098121008	0.147058738	0.199933964	0.267033529	0.365900708	0.647259302
$n = 10$	0.013462706	0.037443215	0.060493049	0.09453266	0.128190148	0.171256984	0.211785472	0.28246711	0.383560047	0.662128031
$n = 11$	0.012043846	0.032981091	0.060959325	0.08094079	0.11207601	0.140879007	0.183428824	0.236724625	0.296740729	0.406202434	0.709898851
$n = 12$	0.01213269	0.031902767	0.053669014	0.074933022	0.099221571	0.123227893	0.160375684	0.20059799	0.247863248	0.315437122	0.424375489	0.669852747
$n = 13$	0.013620096	0.029764391	0.046585371	0.064200709	0.0867847	0.110830852	0.142389726	0.173888317	0.218039963	0.265477127	0.338932811	0.433761148	0.775019978
$n = 14$	0.012535998	0.028136808	0.044420612	0.058894046	0.079973479	0.101200355	0.130906165	0.156937343	0.189795054	0.228742732	0.275643553	0.3453428	0.455101098	0.766034131
$n = 15$	0.010086111	0.028183942	0.04167849	0.058031139	0.075949744	0.095368586	0.113319452	0.145130191	0.171882286	0.204963228	0.240448068	0.296341577	0.360691724	0.469341279	0.755826551
$n = 16$	0.010270239	0.022620476	0.034818048	0.051626665	0.069684203	0.087366547	0.105946087	0.130865469	0.154681334	0.178178489	0.217774223	0.262553766	0.308866451	0.377041142	0.47736129	0.75981994
$n = 17$	0.005957797	0.016174743	0.0311057	0.049640162	0.06682942	0.079542398	0.099743579	0.117661096	0.141725444	0.163728143	0.195581708	0.227925432	0.26764077	0.316660513	0.393807708	0.490684125	0.730301114
$n = 18$	0.010250038	0.018472125	0.031510072	0.045927537	0.059738263	0.076309728	0.092740046	0.107573723	0.131073067	0.152539914	0.176058888	0.20532953	0.238672188	0.282208705	0.325305367	0.399443932	0.502224313	0.777989666
$n = 19$	0.008397473	0.019218156	0.029244322	0.042569472	0.054789402	0.070089018	0.084408128	0.101424034	0.122683274	0.142601758	0.161718117	0.185905197	0.214116541	0.251346118	0.289333816	0.335033328	0.406236006	0.511582141	0.777584673
$n = 20$	0.009582411	0.019302949	0.027503199	0.039537566	0.050284385	0.064794724	0.08022531	0.09729349	0.112747674	0.13148693	0.150876728	0.170319096	0.197338867	0.226214537	0.261230125	0.297500604	0.348479863	0.416432845	0.51940036	0.824544369
$n = 21$	0.00801785	0.015098184	0.02843174	0.038481188	0.048280066	0.061200373	0.074300082	0.089612986	0.105372274	0.123209453	0.139791786	0.160483634	0.181837936	0.208008261	0.2366335	0.267863637	0.30935915	0.362425028	0.430334534	0.534693867	0.830129338
$n = 22$	0.00630777	0.013986447	0.024352912	0.036910559	0.047410421	0.057727229	0.069891649	0.083780568	0.097050695	0.113275499	0.12924367	0.150907577	0.171804212	0.189569326	0.211831916	0.245127456	0.281929854	0.322104817	0.365405287	0.43605103	0.539170156	0.834091009
$n = 23$	0.004000844	0.015765031	0.024906235	0.035327477	0.044038204	0.055009319	0.067244059	0.078594942	0.094726177	0.105740835	0.123142627	0.141738733	0.155288958	0.177395611	0.199746894	0.224104351	0.255900338	0.286384529	0.328100144	0.381246485	0.444025958	0.549014184	0.792496665
$n = 24$	0.006032677	0.013885981	0.021443897	0.033463534	0.040562283	0.053685352	0.063977288	0.074443724	0.08695018	0.101919244	0.116975295	0.1313755	0.147829812	0.16612009	0.185331644	0.210586737	0.232099869	0.265784017	0.297704635	0.33469992	0.387186813	0.455114164	0.557683662	0.810560632
$n = 25$	0.005557423	0.011770911	0.021696757	0.031849098	0.039614443	0.05006037	0.061369932	0.071341087	0.085485478	0.096617797	0.106534246	0.124874047	0.140567436	0.156939326	0.174649627	0.192443408	0.217971963	0.237923558	0.272061375	0.303349244	0.341178886	0.393064331	0.465546983	0.563830808	0.825467782
$n = 26$	0.005497555	0.012894992	0.020411329	0.02743744	0.040719514	0.047987109	0.058526665	0.068677648	0.080667626	0.09144698	0.103722316	0.118415989	0.131512886	0.145928092	0.163277669	0.18232537	0.203268347	0.224734158	0.250796931	0.277003389	0.311064748	0.351599016	0.40571245	0.466943446	0.572954899	0.84286978
$n = 27$	0.005856316	0.01325734	0.019593074	0.026793113	0.036016144	0.04796175	0.056383232	0.066593083	0.077610634	0.087916085	0.098055658	0.112816755	0.125165159	0.140373614	0.155477272	0.171749247	0.189193392	0.209121871	0.229453298	0.260000804	0.285384311	0.319356077	0.359842528	0.409458967	0.475301481	0.583154539	0.853688779
$n = 28$	0.00219039	0.011749541	0.018698569	0.027273642	0.035063754	0.046200872	0.05417457	0.062893188	0.073285331	0.083583538	0.094686634	0.104044467	0.117857601	0.131495246	0.146123157	0.162717303	0.177262081	0.196049755	0.218629425	0.239525553	0.263682245	0.295610513	0.325719045	0.366809362	0.416249814	0.480190228	0.586079889	0.859282238
$n = 29$	0.004622913	0.011210919	0.019242019	0.026938711	0.033516427	0.042676469	0.050165391	0.060567293	0.070573509	0.081530958	0.089440236	0.100142026	0.114810688	0.125919934	0.140324406	0.152694424	0.168515108	0.185719983	0.204057276	0.224493434	0.245739516	0.269597811	0.300660734	0.33404101	0.375604454	0.428136883	0.488998112	0.597451017	0.831293281
$n = 30$	0.00484999	0.0122273	0.019019548	0.025316822	0.03161571	0.03903407	0.048275013	0.057854555	0.067663398	0.077722357	0.086197229	0.097848602	0.108882472	0.119683238	0.133368982	0.145725739	0.16222734	0.175641893	0.192160749	0.211318294	0.231044242	0.253737038	0.277753809	0.307956449	0.340388055	0.381090412	0.428918693	0.496093519	0.602786492	0.8777722

Table 4: OSF results for $f_1 \sim f_3$

Table 5: OSF results for $f_4 \sim f_6$.

Bibliography

[1] Bates, R.A., R.F. Buck, E. Riccomagno and H.P. Wynn (1996). Expimental design and observation for large systems, *J. Roy. Statist. Soc. Ser. B*, 58, 77-94. [[[*this is an example of reference style for Journal ariticles.*]]]

[2] Wang, S, Generale, A. P, Kalidindi, S. R. and Joseph, V. R. (2023). Sequential Designs for Filling Output Spaces. *arXiv preprint arXiv:2305.07202*.

[3] Lu, L. and Anderson-Cook, C. M. (2021). Input-response space-filling designs. *Quality and Reliability Engineering International*, 37(8), 3529-3551.

[4] Viana, F. A, Venter, G. and Balabanov, V. (2010). An algorithm for fast optimal Latin hypercube design of experiments. *International journal for numerical methods in engineering*, 82(2), 135-156.

[5] Krishna, A., Craig, S. R., Shi, C. and Joseph, V. R. (2022). Inverse design of acoustic metasurfaces using space-filling points. *Applied Physics Letters*, 121(7), 071701.

[6] Banach, S. (1922). Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta mathematicae*, 3(1), 133-181.

[7] McKay, M. D., Beckman, R. J. and Conover, W. J. (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1), 55-61.

[8] Krishna, A., Tran, H., Huang, C., Ramprasad, R. and Joseph, V. R. (2023). Adaptive exploration and optimization of materials crystal structures. *INFORMS Journal on Data Science.*

[9] Jourdan, A. (2021). Space-filling designs with a Dirichlet distribution for mixture experiments.

[10] Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(9).

[11] DiCiccio, T. J. and Efron, B. (1996). Bootstrap confidence intervals. *Statistical science,* 11(3), 189-228.

[12] Yule, G. U. and Kendall, M. G. (1911). An introduction to the theory of statistics. *Charles Griffin and Company. London UK*, 147-148.

[13] Pearson, K. (1901). On Lines and Planes of Closest Fit to Systems of Points in Space, *Philosophical Magazine*, 2 (11): 559–572. http://doi.org/10.1080/14786440109462720.

[14] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6), 417.

[15] Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28 (3/4),321–377. http://doi.org/10.2307/2333955.

[16] Elsawah, A. M., Wang, Y. A., Celem, S. M. and Qin, H. (2023). A novel technique for constructing nonregular nine-level designs: Adjusted multiple tripling technique. *Journal of Computational and Applied Mathematics*, 424, 115016.

[17] Gnedenko, B. V., Kolmogorov, A. N., Doob, J. L. and Hsu, P. L. (1968). Limit distributions for sums of independent random variables (Vol. 233). *MA: Addison-wesley.*

[18] Duong, T., & Hazelton, M. L. (2005). Convergence rates for unconstrained bandwidth matrix selectors in multivariate kernel density estimation. *Journal of Multivariate Analysis*, 93(2), 417-433.

[19] Cohen, Irwin., and Bustard, Thomas. (1966). Atomic orbitals: Limitations and variations. *Journal of Chemical Education,* 43(4), 187.www.doi.org/10.1021/ed043p187

[20] Cao, Y., Liang, J., and Gao, N. (2022). A new method for identifying significant genes from gene expression data. *Biom Biostat Int J,* 11(4), 140-146.

[21] Joseph, V. R. (2016). Space-filling designs for computer experiments: A review. *Quality Engineering,* 28(1), 28-35.

[22] Cao, Y., Liang, J., Gao, N., and Sun Z. (2023). Multiple Mean Comparison for Gene Expression Data via F -Type Tests under High Dimension with A Small Sample Size. *American Journal of Biomedical Science and Research,* 18(3), 306-314.

[23] Elsawah, A. M., Wang, Y. A., and Tank, F. (2024). Minimum energy representative points. *Journal of Computational and Applied Mathematics,* 438, 115526.

[24] Newman, D. (1939). The distribution of range in samples from a normal population, expressed in terms of an independent estimate of standard deviation. *Biometrika,* 31(1/2), 20-30.

[25] Gumbel, E. J. (1947). The distribution of the range. *The Annals of Mathematical Statistics,* 18(3), 384-412.

[26] Feller, W. (1951). The asymptotic distribution of the range of sums of independent random variables. *The annals of mathematical statistics,* 427-432.

[27] Wallace, D. L. (1958). Asymptotic approximations to distributions. *The Annals of Mathematical Statistics,* 29(3), 635-654.

[28] Snapp, R. R., and Venkatesh, S. S. (1998). Asymptotic expansions of the k nearest neighbor risk. *The Annals of Statistics,* 26(3), 850-878.

[29] Julier, S. J., and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3), 401-422.

[30] Jaffer, A. (2014). Recurrence for Pandimensional Space-Filling Functions. *arXiv preprint* arXiv:1402.1807.

[31] Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26, 131–148.

[32] Yang, L., Zhou, Y., and Liu, M. Q. (2021). *Maximin distance designs based on densest packings. Metrika*, 84(5), 615-634.

[33] Fang, K., Zhou, M., and Wang, W. (2014). Applications of the representative points in statistical simulations. *Science China Mathematics*, 57, 2609-2620.

[34] Jiang, J. J., He, P., and Fang, K. T. (2015). An interesting property of the arcsine distribution and its applications. *Statistics & Probability Letters*, 105, 88-95.

[35] Xu, L. H., Fang, K. T., and He, P. (2022). Properties and generation of representative points of the exponential distribution. *Statistical Papers*, 1-27.

[36] Li, Y., Fang, K. T., He, P., and Peng, H. (2022). Representative points from a mixture of two normal distributions. Mathematics, 10(21), 3952.

[37] Elsawah, A. M., Wang, Y. A., and Tank, F. (2023). Minimum energy representative points. Journal of Computational and Applied Mathematics, 438, 115526.