

A dependent sampling approach to Scenario Discovery

Master thesis submitted to Delft University of Technology
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in **Engineering and Policy Analysis**

Faculty of Technology, Policy and Management

by

Irene van Droffelaar

Student number: 4365461

To be defended in public on August 26, 2020

Graduation committee

Chairperson : Dr. J.H. Kwakkel, Policy Analysis
First Supervisor : Dr. J.H. Kwakkel, Policy Analysis
Second Supervisor : Dr. M.E. Warnier, Systems Engineering
External Supervisor : Dr. N.Y. Aydin, Systems Engineering

A DEPENDENT SAMPLING APPROACH TO SCENARIO DISCOVERY

A thesis submitted to the Delft University of Technology in partial fulfillment
of the requirements for the degree of

Master of Science in Engineering and Policy Analysis

by

Irene Sophia van Droffelaar

August 2020

An electronic version of this thesis is available at
<http://repository.tudelft.nl/>.

Associated code and models are available at
https://github.com/irene-sophia/MCMC_SD.

Irene Sophia van Droffelaar: *A dependent sampling approach to Scenario Discovery* (2020)

The work in this thesis was made in the:



Policy Analysis
Multi-Actor Systems
Faculty of Technology, Policy & Management
Delft University of Technology

Supervisors: Dr.ir Jan Kwakkel
Dr. Nazli Yonca Aydin
Dr. Martijn Warnier

SUMMARY

The use of scenario planning has a long history in decision-making and public policy (Bryant and Lempert, 2010). Traditional scenario planning, as, for example, used by the Shell Scenarios group, provides tools to communicate and characterize uncertainty, allowing decision-makers to anticipate the future and create more robust strategies (Bradfield et al., 2005).

However, the classical qualitative approach, where scenario narratives are developed, has some severe limitations. While this approach provides results which are readily communicated to decision-makers, it often overlooks truly unexpected (but plausible) scenarios (Kwakkel and Cunningham, 2016). Besides, they are not readily implemented for problems where the structure is also disputed (Bryant and Lempert, 2010).

Scenario discovery, a quantitative, brute-force approach to scenario development, developed by Bryant et al. (2010) aims to address these limitations. This approach has been successfully applied to numerous grand challenges, among others climate change (Kwadijk et al., 2010), sustainable water management (Haasnoot et al., 2011), and global natural resource management (Kwakkel et al., 2013).

The original approach to quantitative scenario discovery relies on three consecutive steps: sampling, labeling and searching for subspaces leading to the regions of interest. The sample is typically collected using Latin Hypercube Sampling (Kwakkel, 2017). For uniform, independent sampling, the chance that a random sample falls within the region of interest decreases quadratically with increasing size of the bounds (Vrugt, 2016). Therefore, if the prior distribution cannot be estimated, or the problem at hand demands a wide range of possible values, and the problem has a high number of dimensions, these sampling techniques require an unreasonably high number of samples to adequately delineate the region(s) of interest.

Therefore, this thesis proposes using *dependent* sampling, which concentrates the sampling on the regions of interest, rather than attempting to represent the entire input space. After convergence, the resulting sample approximates the true posterior distribution. By performing a Kolmogorov-Smirnov test for uniformity, the uncertain factors leading to the behavior of interest can be derived. The relevant ranges of parameter values can be recognized and communicated to decision-makers.

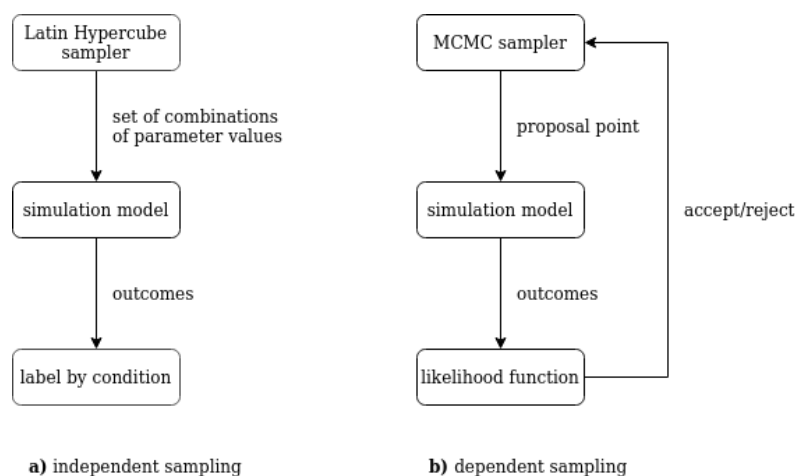


Figure 0.1: Process of generating samples for Scenario Discovery using an a) independent, and b) dependent sampling approach.

PREFACE

To boldly go where no man has
gone before

Captain James T. Kirk

I would hereby like to express my gratitude to my graduation committee for inspiring and supporting me, not only during the process of this thesis, but throughout the Master's. Jan Kwakkel for his contagious enthusiasm and his patience with my process. Nazli Aydin for her support: her input helped me connect the theoretical insights in this thesis to real world applications. Finally, Martijn Warnier for his positive outlook and experienced advise.

Furthermore, I would like to thank my family for their limitless patience when I dedicated yet another call to the intricacies of my thesis. And, of course, the friends without whose nachos, hazelnut lattes and adventures I couldn't have done it.

Irene van Droffelaar
Den Haag, August 2020

CONTENTS

1	INTRODUCTION	2
1.1	Decision support	2
1.2	Research gap	2
1.3	Research questions	3
1.4	Research flow	3
2	RELATED WORK	4
2.1	Scenario Discovery	4
2.1.1	Sampling	4
2.1.2	Classifying outcomes	4
2.1.3	Finding subspaces of interest	5
2.1.4	Limitations	6
2.2	Markov chain Monte Carlo	6
2.3	Approximate Bayesian Computation	8
3	RESEARCH METHODS	10
3.1	Test shapes	10
3.1.1	Simple test problems	10
3.1.2	High-dimensional, sparse test problems	10
3.2	Algorithms	12
3.2.1	Latin Hypercube Sampling	13
3.2.2	Adaptive Metropolis	13
3.2.3	Differential Evolution Markov Chain	14
3.2.4	Differential Evolution Adaptive Metropolis	15
3.2.5	Shape of the likelihood function	18
3.3	Performance measures	20
3.3.1	Interpretability	21
3.3.2	Quality	21
3.3.3	Efficiency	21
3.3.4	Selection of measures for research	23
4	VISUALIZATION AND COMMUNICATION OF RESULTS	24
4.1	Independent sampling	24
4.1.1	Applicability of original approach to MCMC sample	24
4.2	Dependent sampling	26
4.2.1	Deriving the dimensions of interest	26
4.2.2	Regions of interest	28
4.2.3	Demonstration for three cases	31
4.3	Conclusions	34
5	COMPARING ALGORITHMS	35
5.1	Experimental setup	35
5.1.1	Test shapes	35
5.1.2	Algorithms	35
5.2	Latin Hypercube Sampling	36
5.2.1	Simple test problems	36
5.2.2	High-dimensional test problems	36
5.3	Adaptive Metropolis	37
5.3.1	Simple test problems	37
5.3.2	High-dimensional test problems	38
5.4	Differential Evolution Markov Chain	38
5.4.1	Simple test problems	38
5.4.2	High-dimensional test problems	40
5.5	Differential Evolution Adaptive Metropolis	41

5.5.1	Simple test problems	41
5.5.2	High-dimensional test problems	41
5.6	Comparing algorithms	43
5.7	Conclusions	44
6	CONCLUSIONS & DISCUSSION	45
6.1	Answering the research question	45
6.1.1	Proposed methodology	46
6.2	Discussion	47
6.2.1	Predetermined definitions of behavior of interest	47
6.2.2	Computational considerations	47
6.2.3	Comparing computational efficiency	48
6.2.4	Discrete and categorical parameters	48
6.2.5	Finding all regions of interest	49
6.3	Impact of research	49
6.4	Future directions of research	49
A	THE SHAPE OF THE LIKELIHOOD FUNCTION	53
A.1	Adaptive Metropolis	53
A.1.1	Simple test problems	53
A.1.2	High-dimensional test problems	53
A.2	Differential Evolution Markov Chain	54
A.2.1	Simple test problems	54
A.2.2	High-dimensional test problems	54
A.3	Differential Evolution Adaptive Metropolis	56
A.4	Choosing the likelihood function	56

LIST OF FIGURES

Figure 2.1	Sampling coverage of pseudo-random (factorial and latin hypercube) and random sampling methods in 2D. Adapted from (Weber, 2019).	4
Figure 2.2	The relationship between the model outcomes, the threshold and the shape of the likelihood landscape. This likelihood function follows some sigmoidal shape, but any shape that fits the problem at hand is allowed.	7
Figure 2.3	Process of generating samples for Scenario Discovery using an a) independent, and b) dependent sampling approach.	7
Figure 2.4	Conceptual overview of Approximate Bayesian Computation for a one-dimensional parameter estimation problem. Adapted from Sadegh and Vrugt (2014); Sunnåker et al. (2013).	8
Figure 3.1	3d plots of the Latin Hypercube sampling of disjoint barrels in 6d	12
Figure 3.2	Histogram of the posterior distribution derived from (a) AM and (b) DE-MC. Reprinted from "Markov chain Monte Carlo simulation using the DREAM soft-ware package: Theory, concepts, and MATLAB implementation" by Vrugt, J.A., 2016, Environmental Modeling and Software, 75, 273–316.	15
Figure 3.3	From Vrugt (2016): DREAM (ZS) algorithm: Explanation of the snooker update for a hypothetical two-dimensional problem using some external archive of $m = 10$ points (grey dots). Three points of this archive \mathbf{Z}^a , \mathbf{Z}^b and \mathbf{Z}^c are sampled at random and define the jump of the i th chain, \mathbf{X}^i (blue) as follows. The points \mathbf{Z}^b and \mathbf{Z}^c are projected orthogonally on to the dotted $\mathbf{X}^i\mathbf{Z}^a$ line. The jump is now defined as a multiple of the difference between the projections points, \mathbf{Z}_\perp^b and \mathbf{Z}_\perp^c (green squares) and creates the proposal, \mathbf{X}_p^i . The DREAM (ZS) algorithm uses a 90/10 % mix of parallel direction and snooker updates, respectively. . .	17
Figure 3.4	Graphs of the examined likelihood functions. The grey vertical line depicts the boundary of the region of interest. The returned value of the likelihood function (y-axis) decreases with increasing distance from this boundary.	20
Figure 3.5	Pairplots of the Adaptive Metropolis algorithm (using the Metropolis Ratio for acceptance/rejection) applied to the 'upright barrel shape' with nfe=10,000, for various likelihood functions. The region of interest is indicated in red.	22
Figure 4.1	PRIM performed on LHS (independent sampling) results for the 50d, small test problem. Orange dots are cases of interest, blue dots are not.	25
Figure 4.2	Illustration of the Kolmogorov–Smirnov test for a normal distribution, with the cumulative density function of the sample in blue and the reference distribution in red. The pairwise distances between the sample and the reference distribution are calculated at each point. The maximum of these distances is the recorded statistic. Adapted from Mathworks (2020).	27
Figure 4.3	Histograms of the distribution of samples for two dimensions: one which does and one that does not contribute to the region of interest; to illustrate how the difference may be distinguished from an MCMC sample. The orange line is a uniform distribution.	27

Figure 4.4	Cumulative probability functions of x_1 to x_{50} in blue. Dimensions with a KS statistic of > 0.1 (i.e. x_1, x_2, x_{48} and x_{49}) are coloured orange for clarity. The cumulative probability of a uniform distribution is displayed in red.	27
Figure 4.5	Barplot of the value of the KS statistic for each of the dimensions in the DREAM(ABC) sample of the 50d, small test problem with nfe=30,000. The KS statistic is calculated by comparing to a uniform distribution with bounds $[-0.5, 0.5]$	28
Figure 4.6	Pairplot of the selected dimensions of the DREAM(ABC) sample of the 50d, large test problem.	29
Figure 4.7	Probability density for each dimension of interest for the large, 50d test problem, generated by kernel density estimation. The ranges of parameter values are indicated by the vertical orange lines.	30
Figure 4.8	Dimension of interest x_1	31
Figure 4.9	Dimension of interest x_2 , given $-0.20 < x_1 < 0.18$	32
Figure 4.10	Barplot of the value of the KS statistic for each of the dimensions in the DREAM(ABC) sample of the 50d, large test problem with nfe=30,000. The KS statistic is calculated by comparing to a uniform distribution with bounds $[-5, 5]$	33
Figure 5.1	Colour-coded pairplots, generated from a Latin Hypercube sample of size 10000.	37
Figure 5.2	Performance of the AM algorithm for the simple test problems	37
Figure 5.3	The AM algorithm has no trouble finding both input spaces in the small, 6d problem.	38
Figure 5.4	The starting position of the AM algorithm is in a corner of the 50-dimensional, large input space, leading to a long trajectory to the region of interest (in the middle of the input space). The number of function evaluations proves insufficient for the algorithm to converge to the region of interest. Only the three dimensions defining the first region of interest are presented here.	39
Figure 5.5	Histograms of the value of the KS statistic for each of the dimensions in the Adaptive Metropolis sample of the 50d, large test problem. The single-chain method only finds one region of interest, because the jumping distance is insufficient to find both.	39
Figure 5.6	Performance of the DE-MC algorithm for the simple test problems	40
Figure 5.7	Histograms of the value of the KS statistic for each of the dimensions in the Differential Evolution Markov Chain sample of the 50d, large test problem.	40
Figure 5.8	Cumulative probability functions of x_1 to x_{50} (DE-MC, nfe=30,000). Dimensions $x_1, x_2, x_3, x_{47}, x_{48}$ and x_{49} are coloured orange for clarity. The cumulative probability of a uniform distribution is displayed in red.	40
Figure 5.9	Performance of the DREAM algorithm for the simple test problems	41
Figure 5.10	DREAM(ABC) sample of 50d test problem, with large bounds	42
Figure 5.11	Histograms of the value of the KS statistic for each of the dimensions in the DREAM(ABC) sample of the 50d, large test problem. The KS statistic is calculated by comparing to a uniform distribution with bounds $[-5, 5]$	42
Figure 5.12	Acceptance rates and fractions of interest of the various algorithms for various test shapes in 3d, for 1000 and 30,000 nfe	43

Figure 5.13	Acceptance rates and fractions of interest of the various algorithms for various high-dimensional test shapes, for 1000 and 30,000 nfe	44
Figure A.1	Performance of the Adaptive Metropolis algorithm for various likelihood functions	53
Figure A.2	Performance of the AM algorithm for various likelihood functions for the 50d, large box test problem	54
Figure A.3	Performance of the DE-MC algorithm for various likelihood functions	55
Figure A.4	Performance of the DE-MC algorithm for various likelihood functions for the 50d, large box test problem	55
Figure A.5	Performance of the DREAM(ABC) algorithm for various likelihood functions	56

LIST OF TABLES

Table 3.1	Barrel shapes from Lempert et al. (2008) and their respective equations	11
Table 3.2	High dimensional test shapes and their respective equations (adapted from Lempert et al. (2008))	12
Table 3.3	Input to various algorithms to guarantee the same number of function evaluations	22
Table 4.1	Ranges of values for uncertain parameters defining the regions of interest, for LHS sample of 50d, small test problem using PRIM.	24
Table 4.2	Ranges of parameter values delineating the region of interest for the 50d, small test problem	30
Table 4.3	KS statistic for x_1, x_2, x_3 , given the conditions found in each step.	32
Table 4.4	First round of the process: KS statistic for x_1, x_2, x_3 , given the conditions found in each step.	32
Table 4.5	Second round of the process: KS statistic for x_1, x_2, x_3 , given the conditions found in each step.	33
Table 4.6	Regions of interest found by a PRIM analysis of a Latin Hypercube sample, contrasted with the results of the proposed methodology for dependent sampling.	33
Table 4.7	First round of the process: KS statistic for $x_1, x_2, x_3, x_{47}, x_{48}, x_{49}$ given the conditions found in each step.	34
Table 4.8	Fourth round of the process: KS statistic for $x_1, x_2, x_3, x_{47}, x_{48}, x_{49}$ given the conditions found in each step.	34
Table 5.1	Fractions of interest of LH sample of various barrel shapes in 3d	36
Table 5.2	Fractions of interest of LH sample of various higher dimensional test problems	37

List of Algorithms

3.1	Latin Hypercube Sampling (adapted from Shockley et al. (2017))	13
3.2	Adaptive Metropolis (adapted from Vrugt (2016))	14
3.3	Differential Evolution Markov Chain (adapted from Vrugt (2016))	16
3.4	Differential Evolution Adaptive Metropolis (adapted from Vrugt (2016))	19
3.5	Likelihood function - generic	19

1

INTRODUCTION

1.1 DECISION SUPPORT

When dealing with important policy problems, policymakers are faced with high levels of uncertainty, where a stochastic approach is insufficient (Walker et al., 2013). More specifically, this *deep uncertainty* is defined by Lempert et al. (2003) by three criteria. The following are unknown or cannot be agreed on:

1. the conceptual model describing the relationships between factors.
2. the probability distributions of the factors used to represent the uncertainty in the model
3. the outcomes of interest and their relative valuation

The use of scenario planning has a long history in decision-making and public policy (Bryant and Lempert, 2010). Traditional scenario planning, as, for example, used by the Shell Scenarios group, provides tools to communicate and characterize uncertainty, allowing decision-makers to anticipate the future and create more robust strategies (Bradfield et al., 2005).

However, the classical qualitative approach, where scenario narratives are developed, has some severe limitations. While this approach provides results which are readily communicated to decision-makers, it often overlooks truly unexpected (but plausible) scenarios (Kwakkel and Cunningham, 2016). Besides, they are not easily implemented for problems where the model structure is also disputed (Bryant and Lempert, 2010).

Scenario discovery, a quantitative, brute-force approach to scenario development aims to address these limitations (Groves and Lempert, 2007). This approach has been successfully applied to numerous grand challenges, among others climate change (Kwadijk et al., 2010), sustainable water management (Haasnoot et al., 2011), and global natural resource management (Kwakkel et al., 2013).

1.2 RESEARCH GAP

Scenario Discovery depends on a representative sample of the input space for analysis. However, for uniform, independent sampling, the chance that a random sample falls within the region of interest decreases quadratically with increasing size of the bounds (Vrugt, 2016). Therefore, if the prior distribution cannot be estimated, or the problem at hand demands a wide range of possible values, and the problem has a high number of dimensions, these sampling techniques require an unreasonably high number of samples to adequately delineate the region(s) of interest. Therefore, this thesis proposes using *dependent* sampling, which concentrates the sampling on the regions of interest, rather than attempting to represent the entire input space.

More specifically, this is achieved by using Markov Chain Monte Carlo (MCMC) methods with multiple chains to direct the sampling to subspaces of interest. This approach should result in a significantly higher number of cases of interest. A risk is that these algorithms do not recognize other disconnected regions of interest. Therefore, the algorithms are thoroughly tested and examined by performing experiments on the known, simple shapes presented by Lempert et al. (2008).

1.3 RESEARCH QUESTIONS

The following main research question was formulated to address the research gap previously described:

How can dependent sampling methods be used for effective Scenario Discovery to characterize high-dimensional problems with sparse regions of interest?

In order to answer this research question, firstly a literature review will be conducted to select appropriate algorithms and measures of performance. These two objectives form the first two research questions:

1. *Which dependent sampling algorithms may be employed for Scenario Discovery?*
2. *Which measures of performance can be used to compare various algorithms for Scenario Discovery?*

Secondly, an appropriate way to process an MCMC sample for visualization and communication will be designed. This is also a proof of concept of the applicability of independent sampling to Scenario Discovery.

3. *How can combinations of input parameters that are highly predictive of the behavior of interest be derived from an MCMC sample?*

Finally, a number of algorithms (selected in subquestion 1) are applied to increasingly challenging (but still known) problems, to study their relative performance on high-dimensional problems with sparse regions of interest with regards to the goals of Scenario Discovery. This forms the fourth research question:

4. *What is the relative efficacy of dependent sampling algorithms for high-dimensional problems with sparse regions of interest, compared to an independent sampling approach?*

Together, these subquestions address the main question posed previously.

1.4 RESEARCH FLOW

This thesis consists of six chapters. This first chapter introduces the topic of the thesis and discusses the research gap addressed by the research project. A number of subquestions are formulated to guide the research. Then, the principles of Scenario Discovery and dependent sampling algorithms are presented in the literature review. This chapter provides answers to the first subquestion. Subsequently, the experimental setup and research methods will be discussed, including the performance measures used to compare approaches to Scenario Discovery, answering the second subquestion. The following chapter discusses a method to derive the combinations of input parameters that lead to the behavior of interest from an MCMC sample. The subsequent chapter presents increasingly challenging test cases for algorithms for Scenario Discovery to answer the fourth subquestion. The final chapter discusses the main research question and reflects on the research study.

2 | RELATED WORK

2.1 SCENARIO DISCOVERY

Scenario Discovery is a quantitative approach to find the combinations of input parameter values (or ‘uncertainties’) that lead to an outcome of interest (Bryant and Lempert, 2010). The approach includes three steps (Lempert et al., 2008), which will each be discussed in more detail in the subsections below.

1. Sample parameter values over prior distributions. These values are used as inputs to a simulation model and the resulting outcomes are recorded.
2. Classify outcomes as ‘of interest’ or ‘not of interest’, based on some user-defined condition.
3. Use Machine Learning techniques to search for the combinations of ranges of parameter values causing the behavior of interest (Groves and Lempert, 2007).

These combinations of uncertain model input parameters that are predictive of policy-relevant cases are translated into scenarios, which are used to inform policy-makers. This provides a tool for decision-making in situations of deep uncertainty (Groves and Lempert, 2007).

2.1.1 Sampling

Sampling methods typically used in this field are *independent*, and include fully random methods (Kim et al., 2000), Latin Hypercube Sampling (pseudo random) (McKay et al., 1979), and factorial (pseudo random) (Fisher, 1935) methods. These sampling methods are independent (i.e. sampling does not depend on other observations) and, given sufficient samples, guarantee the coverage of the full parameter space. The differences between these three methods are illustrated in figure 2.1, showing samples from a 2D space.

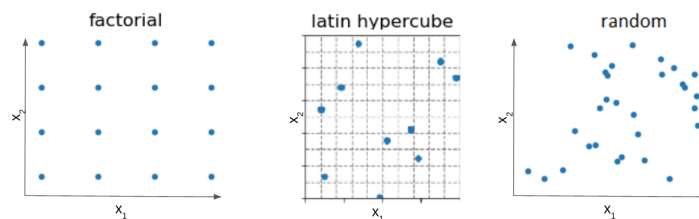


Figure 2.1: Sampling coverage of pseudo-random (factorial and latin hypercube) and random sampling methods in 2D. Adapted from (Weber, 2019).

2.1.2 Classifying outcomes

‘Cases of interest’ are often defined by some threshold for ‘failure’ or ‘success’, but could be defined as any threshold, value, or range of values for a certain outcome, or a combination of outcomes (Lempert et al., 2006). Scenarios of interest could even be labeled by their behavior over time (Steinmann et al., 2020).

2.1.3 Finding subspaces of interest

Finally, some algorithm is employed to identify subspaces in the input space that explain the outcomes of interest. While various algorithms may be applied, algorithms typically used in this field are Classification And Regression Trees (CART) and the Patient Rule Induction Method (PRIM) (Lempert et al., 2008), of which the latter is more often used (Kwakkel, 2019). PRIM and CART balance three performance measures: interpretability, coverage and density, which together maximize the explanatory power of the boxes (Lempert et al., 2008).

Interpretability

Firstly, the interpretability of the results is considered, which is quantified by restricting the number of boxes to three or four, with each a maximum of two to three restricted dimensions.

Coverage

Coverage refers to the proportion of relevant cases, i.e. the ratio of cases of interest *within* a box to the total number of cases of interest in the dataset. A coverage of 1 indicates that the box contains all of the cases of interest (Kwakkel and Cunningham, 2016).

Density

The third performance measure proposed by Lempert et al. (2008) is density, which refers to the fraction of cases of interest within the box set, relative to the total number of cases in that box set. It reflects whether a box primarily describes cases of interest (high density), or if there is a lot of noise (low density).

Coverage and density are usually mutually exclusive: decreasing the box size typically decreases the coverage, while increasing the density. For example, the box that contains *all* cases has a coverage of 1, but a very low density.

Patient Rule Induction Method

PRIM starts with a box that encompasses the entire uncertainty space and incrementally decrease the size of the box along one dimension at a time. Each step, the box becomes smaller, containing fewer irrelevant cases. Each step, the density of this subspace is improved, while keeping as many cases of interest as possible (Friedman and Fisher, 1999). Since only one dimension is restricted in any step, the solution subspace has a lower number of dimensions than the full uncertainty space. The results are typically easily interpreted by decision-makers (Lempert et al., 2008). PRIM can be repeated to find additional boxes, explaining different parts of the uncertainty space (Kwakkel and Cunningham, 2016).

Classification And Regression Trees

CART outputs decision trees consisting of splitting criteria that determine the output class based on input parameters (Breiman et al., 1984). In this process, it optimizes for the lowest classification error. The user can choose the tree with the best predictive power, i.e. a good balance between density and coverage on the one side, and complexity on the other. Still, CART typically yields less interpretable results compared to PRIM (Lempert et al., 2008).

2.1.4 Limitations

Scenario Discovery is typically used in combination with models with a relatively low number of uncertain parameters (9 (PRIM) (Bryant and Lempert, 2010), 6 (PRIM) (Moksnes et al., 2019), 9 (PRIM) (Halim et al., 2016), 18 (CART) (Agusdinata, 2008)). Using the classical approach, it is not feasible to create a representative sample of a higher number of uncertain factors in a reasonable time span. Any search on these samples would yield nonsensical results (Vrugt and Beven, 2018).

For uniform, independent sampling, the chance that a random sample falls within the region of interest decreases quadratically with increasing size of the bounds (Vrugt, 2016). If the prior distribution cannot be estimated, or the problem at hand demands a wide range of possible values, and the problem has a high number of dimensions, these sampling techniques require an unreasonably high number of samples to adequately delineate the region(s) of interest. Therefore, this thesis proposes using *dependent* sampling, which concentrates the sampling on the regions of interest, rather than attempting to represent the entire input space.

2.2 MARKOV CHAIN MONTE CARLO

In Markov chain Monte Carlo simulation, proposal samples are generated from the current state of the Markov Chain (Metropolis et al., 1953). This proposed sample is accepted or rejected, dependent on its relative performance using the Metropolis ratio:

$$p_{\text{acc}}(\mathbf{x}_{t-1} \rightarrow \mathbf{x}_p) = \min \left[1, \frac{p(\mathbf{x}_p)}{p(\mathbf{x}_{t-1})} \right] \quad (2.1)$$

where $p(\mathbf{x}_p)$ and $p(\mathbf{x}_{t-1})$ represent the performance of the proposal point \mathbf{x}_p and the current point \mathbf{x}_{t-1} , respectively. If the performance of the proposal is equal to or better than the current point, the proposal is always accepted. If it is worse, it is *sometimes* accepted, with a probability proportional to the relative performance. If the proposed sample is accepted, the chain moves to \mathbf{x}_p , and if it is rejected, the chain remains in \mathbf{x}_{t-1} . After a sufficient number of iterations, the chain represents the posterior density function. This requires a burn-in period, in which the chain explores the search space (Vrugt, 2016).

The performance of a proposed sample is evaluated through a likelihood function. The likelihood is the probability that the observed data is generated by the parameter values to be sampled (θ): $\text{likelihood}(\theta) = f(x_1, \dots, x_n | \theta)$. The likelihood function transforms the raw model outcomes to an energy landscape defining regions of interest, similar to the ‘labeling’ step in Scenario Discovery (section 2.1). This process is graphically explained in figure 2.2, where the left-hand graph is an arbitrary outcome y , given an independent variable x_1 . Performance exceeding the threshold is marked ‘of interest’ and given a likelihood of 1, i.e. points within this region of interest are always accepted. The likelihood function decreases with increasing distance from the region of interest.

The choice of the shape of this function is largely dependent on what the model returns (i.e. a (log-)likelihood, simulation outcomes, or summary statistics). One can then decide on a suitable likelihood function using Table 2 in Vrugt (2016).

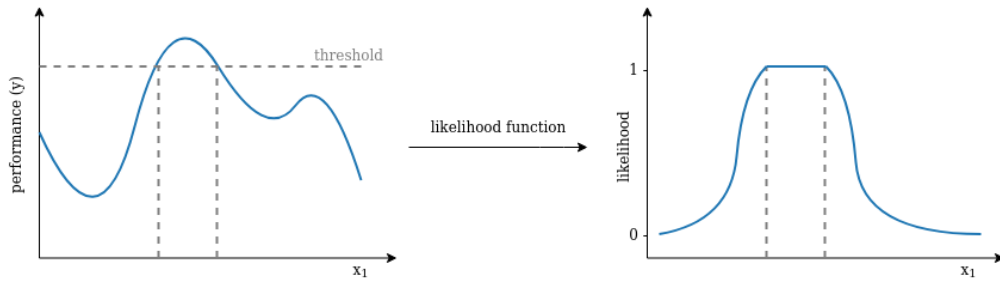


Figure 2.2: The relationship between the model outcomes, the threshold and the shape of the likelihood landscape. This likelihood function follows some sigmoidal shape, but any shape that fits the problem at hand is allowed.

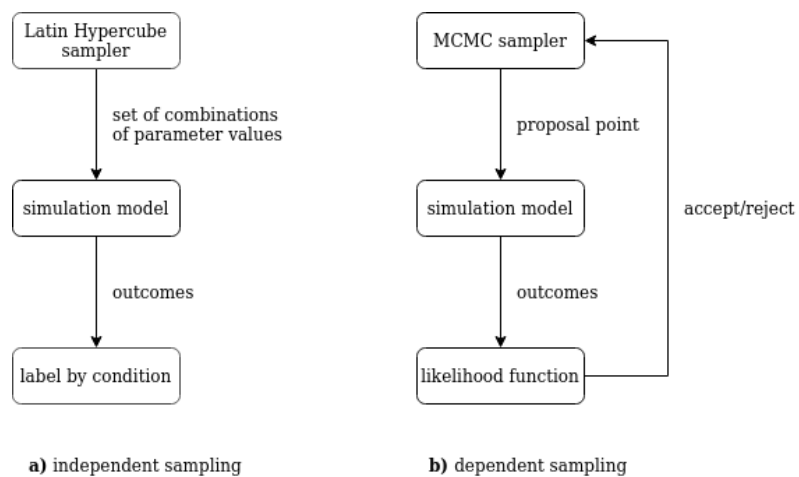


Figure 2.3: Process of generating samples for Scenario Discovery using an a) independent, and b) dependent sampling approach.

2.3 APPROXIMATE BAYESIAN COMPUTATION

MCMC requires the definition of a formal likelihood function (e.g. Table 2 in [Vrugt \(2016\)](#)). However, for the complex problems subjected to Scenario Discovery, it is impossible to formulate a formal likelihood function ([Vrugt and Sadegh, 2013](#)). Approximate Bayesian Computation (ABC) constitutes a class of computational methods often used in model calibration or validation, that avoid this formal likelihood function ([Diggle and Gratton, 1984](#)). Rather, samples are compared to observed data and rejected if the difference between the chosen summary statistics is too large.

Figure 2.4 shows a conceptual overview of ABC for model calibration (its intended purpose). The model ($\frac{dy_t}{dt} = f(y_t, \theta, \tilde{u}^*)$, where y_t is the outcome of interest, θ^* the prior distribution of parameter values and \tilde{u} the observed data, is fitted to the observed data. First, a threshold is chosen for ϵ , which is a trade-off between the accuracy and the computational efficiency. Then, N samples are drawn, using a sampling method of preference. Using this sample, N simulations are ran, collecting some outcome of interest or summary statistic. If the distance between the observed and simulated data is smaller than or equal to ϵ , this sampled point is accepted. The resulting sample, consisting only of accepted points, is used to approximate the posterior parameter distribution $p(\theta|Y)$ ([Sadegh and Vrugt, 2014](#)).

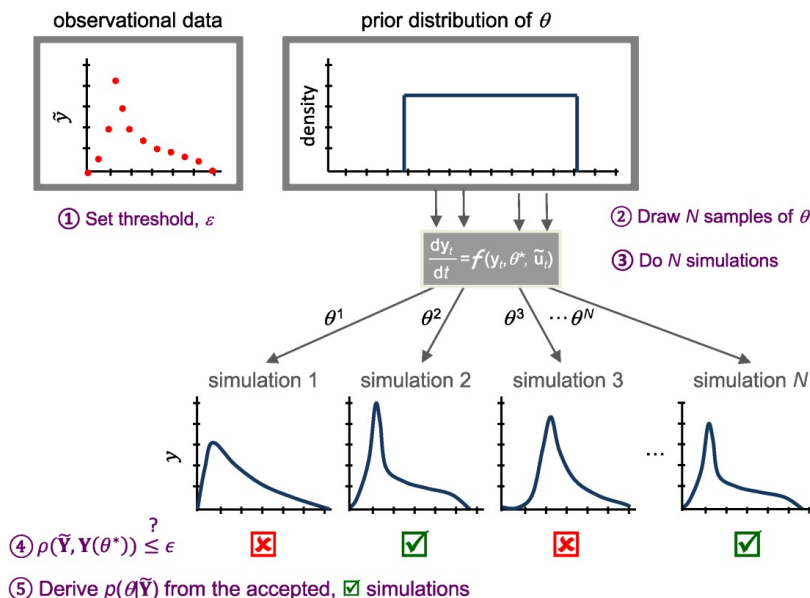


Figure 2.4: Conceptual overview of Approximate Bayesian Computation for a one-dimensional parameter estimation problem. Adapted from [Sadegh and Vrugt \(2014\)](#); [Sunnåker et al. \(2013\)](#).

As previously mentioned, ABC requires a sample of the input space, which can be obtained through any sampling scheme. For large input spaces, it is infeasible to generate a representative sample of the entire input space. Therefore, ABC is often combined with rejection methods like MCMC.

An MCMC implementation of ABC follows these steps ([Sunnåker et al., 2013](#)):

1. Choose one or more summary statistics μ that describe the the observed data well (for example the mean values of various parameters)
2. Generate a candidate point from the current position (note that different MCMC algorithms do this in different ways)
3. Evaluate the model at this proposal point and collect the summary statistic(s)

4. Choose a tolerance value ϵ . If the difference between the observed summary statistics and the simulated summary statistics is smaller than this tolerance value, the candidate point is accepted. If the difference is larger, it is accepted depending on the Metropolis ratio (Turner and Van Zandt, 2012):

$$\alpha = \begin{cases} \min\left(1, \frac{\pi(\theta^*)q(\theta_i|\theta^*)}{\pi(\theta_i)q(\theta^*|\theta_i)}\right) & \text{if } \rho(X, Y) \leq \epsilon_0 \\ 0 & \text{if } \rho(X, Y) > \epsilon_0 \end{cases} \quad (2.2)$$

5. By repeating steps 2-4 n times, the true posterior distribution is approximated (Sadegh and Vrugt, 2014).

Of course, when applying Scenario Discovery, we are not comparing simulated data with observed data. However, this process may be adapted to fit the objectives of Scenario Discovery.

In Scenario Discovery, regions of interest are defined by some threshold or condition, similar to the threshold value ϵ in ABC. However, while in ABC, there is still a difference in performance within this region of interest, in this case we are equally interested in all points within the region of interest. Therefore, the likelihood function is chosen to be uniform *within* the region of interest. Within the region of interest, the ratio will always be 1. Outside the region of interest, it decreases with increasing distance from ϵ , to accommodate for more sparse samples. The chain is able to slowly move towards a region of interest, instead of getting stuck in empty subspaces.

3 | RESEARCH METHODS

3.1 TEST SHAPES

Following the methodology used in (Lempert et al., 2008), the performance of the algorithms for Scenario Discovery is examined using known, simple shapes.

3.1.1 Simple test problems

These different shapes were designed to be relatively easily visualized, while presenting a range of challenges to CART and PRIM (Lempert et al., 2008):

- upright barrel: should not pose any issues; baseline.
- tilted barrel: not orthogonal, so PRIM/ CART may have a hard time drawing a box around the region of interest.
- crossed barrels: even though the shape is originally described by combining two shapes, PRIM and CART need 3 separate boxes to describe it, because the algorithms remove the cases of interest described by the first box, leaving a gap in the middle.
- disjoint barrels: presenting a test problem with two disconnected region of interest forces the algorithms to describe one after the other. PRIM may try to describe disjoint regions with single box.

Similarly, the relative performance of the MCMC methods are examined using these shapes. An important challenge for the MCMC methods are the disjoint barrels, since it is possible for the chains to get stuck in one of the barrels, without exploring the rest of the uncertainty space. The various testshapes are presented in table 3.1. The figures were generated using a 1000-point Latin Hypercube sample, with $-0.5 < x_j < 0.5$ so that the shape is in the middle of the space. PRIM and CART look for orthogonal boxes.

3.1.2 High-dimensional, sparse test problems

Additional experiments were created, where the dimensionality of the uncertainty space is increased significantly, and where the disjoint barrels are in different subspaces. This increases the level of the challenge to the algorithms and mimics the envisioned application of the MCMC algorithms, i.e. a large high-dimensional uncertainty space with small, sparse regions of interest. The barrels are placed in different subspaces, and further apart in the larger test problems, so that the chains do not easily jump from one region of interest to another. This additional challenge examines whether the algorithms are able to find all the regions of interest, instead of getting stuck in local optima.

Latin Hypercube Sampling is expected to perform poorly on these sparse uncertainty spaces, requiring a large number of samples to adequately characterize the regions of interest. Monte Carlo methods risk not finding the regions of interest at all, or getting stuck in one of the regions of interest. The following problems are formulated:

Table 3.1: Barrel shapes from [Lempert et al. \(2008\)](#) and their respective equations

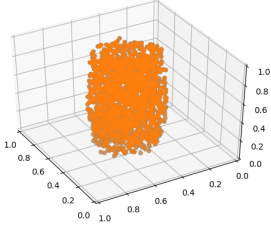
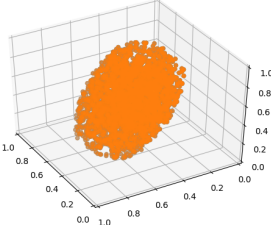
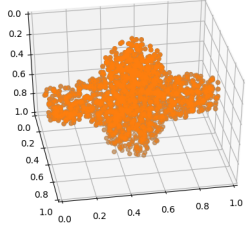
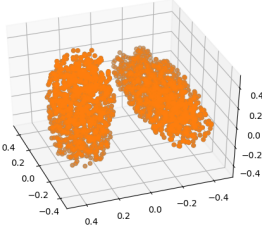
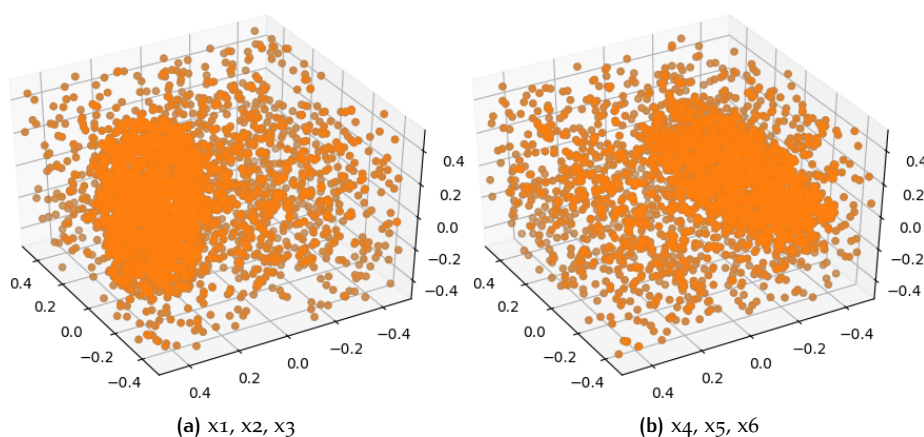
Barrel shape	Equation(s)	Figure
Upright	$16(x_1)^2 + 16(x_2)^2 + (x_3)^2 < 1$	
Tilted	$\frac{25}{4}(x_1)^2 + 16(x_2)^2 + \frac{25}{9}(x_3)^2 + 5(x_1x_2 + x_1x_3 + x_2x_3) < 1$	
Crossed	$25(x_1)^2 + 25(x_2)^2 + \frac{9}{4}(x_3)^2 < 1$ $\frac{9}{4}(x_1)^2 + 25(x_2)^2 + 25(x_3)^2 < 1$	
Disjoint	$25(x_1)^2 + 25(x_2 - 0.25)^2 + \frac{9}{4}(x_3)^2 < 1$ $\frac{9}{4}(x_1)^2 + 25(x_2 + 0.25)^2 + 25(x_3)^2 < 1$	

Table 3.2: High dimensional test shapes and their respective equations (adapted from [Lempert et al. \(2008\)](#))

Dimensions	Bounds	Equations
6	-0.5, 0.5	$25(x_1)^2 + 25(x_2 - 0.25)^2 + \frac{9}{4}(x_3)^2 < 1$ $\frac{9}{4}(x_4)^2 + 25(x_5 + 0.25)^2 + 25(x_6)^2 < 1$
6	-5, 5	$25(x_1 - 2)^2 + 25(x_2 - 2.25)^2 + \frac{9}{4}(x_3 - 2)^2 < 1$ $\frac{9}{4}(x_4 + 2)^2 + 25(x_5 + 2.25)^2 + 25(x_6 + 2)^2 < 1$
50	-0.5, 0.5	$25(x_1)^2 + 25(x_2 - 0.25)^2 + \frac{9}{4}(x_3)^2 < 1$ $\frac{9}{4}(x_{47})^2 + 25(x_{48} + 0.25)^2 + 25(x_{49})^2 < 1$
50	-5, 5	$25(x_1 - 2)^2 + 25(x_2 - 2.25)^2 + \frac{9}{4}(x_3 - 2)^2 < 1$ $\frac{9}{4}(x_{47} + 2)^2 + 25(x_{48} + 2.25)^2 + 25(x_{49} + 2)^2 < 1$

**Figure 3.1:** 3d plots of the Latin Hypercube sampling of disjoint barrels in 6d

Furthermore, the 50-dimensional test problems expose the challenge of visualization and interpretation of the results of MCMC algorithms. The results yield a suitable test case to answer the last sub-question.

The shapes of the high-dimensional test problems are verified by generating two 3d plots of the dimensions that the barrels should be in. These plots (displayed in figure 3.1), show some clutter, since points are of interest because of their parameter values for x_1, x_2 and x_3 , regardless of their values of x_4, x_5 and x_6 (and vice versa).

3.2 ALGORITHMS

For the independent sampling, Latin Hypercube Sampling will be implemented, since this is the most widely used ([Lempert et al., 2006](#)). For the dependent sampling techniques, this thesis follows the line of reasoning by [Vrugt \(2016\)](#), where a number of MCMC methods are compared. Each of the algorithms has a transparent Python implementation. These implementations are adapted to follow the Approximate Bayesian Computation paradigm and record additional statistics, but otherwise implemented out-of-the-box.

The following list describes the algorithms selected for the experiments in this thesis, along with their respective added insights.

- Latin Hypercube Sampling: Contrast dependent sampling techniques with independent sampling technique (reference)
- Adaptive Metropolis: Contrast multi-chain methods with single chain method

- Differential Evolution Markov chain: Contrast DREAM with its predecessor. Most interestingly, DE-MC lacks subspace sampling.
- Differential Evolution Adaptive Metropolis: Examine high-performance MCMC algorithm, enhanced adaptation of DE-MC.

The following sections describe the implementation and settings used for each algorithm for the experiments in this thesis.

3.2.1 Latin Hypercube Sampling

Latin Hypercube Sampling is a stratified approach. It divides each input variable into intervals of equal probability and one sample is drawn from each partition (McKay et al., 1979). These samples for each dimension are randomly combined to an n-dimensional sample. This guarantees that the input space is uniformly sampled given a limited sample size.

Implementation used in experiments

Experimentation and analysis of LHS is performed through the Exploratory Modelling and Analysis (EMA) Workbench (Kwakkel, 2017). This open source python library facilitates the sampling of real, categorical or discrete variables, and the analysis of the samples using PRIM.

Algorithm 3.1: Latin Hypercube Sampling (adapted from Shockley et al. (2017))

Input: minn: lower bound (array of length d)
maxn: upper bound (array of length d)
N: number of samples
Output: A numpy array of length N containing LHS samples

```

1 y = np.random.rand(N, len(minn))
2 x = np.zeros((N, len(minn)))
3 for j in range(len(minn)) do
4     idx = np.random.permutation(N)
5     P = (idx - y[:, j])/N
6     x[:, j] = minn[j] + P * (maxn[j] - minn[j])
7 return x

```

3.2.2 Adaptive Metropolis

Adaptive Metropolis (AM) is an adaptation of the Random Walk Metropolis (RWM) algorithm. The performance of RWM is highly dependent on the choice of the proposal distribution. If it is too wide, many of the resulting proposed samples are rejected, causing slow convergence. If it is too narrow, the distance moved each step is very slow, requiring many samples to converge (Vrugt, 2016). Where RWM uses a static distribution, AM tunes this proposal distribution dynamically using the accepted samples (Haario et al., 2001).

Implementation used in experiments

The implementation of Adaptive Metropolis used in this thesis is based on the one presented in Vrugt (2016), where the MATLAB implementation of DREAM is presented, and its concepts and performance are contrasted with single-chain (AM) and multi-chain (DE-MC) methods. It is adapted to follow the logic of Approximate Bayesian

Computation, which avoids the use of a formal likelihood function. Rather, the posterior is evaluated directly. In the algorithm (3.2), this mainly affects line 9, where the proposed sample is accepted or not. In a classical Metropolis sampling algorithm, acceptance is based on the Metropolis ratio, which is the fraction of the likelihood of the proposal over the likelihood of the previous (accepted) point. If this ratio is larger or equal to u (from $U[0,1]$), the proposal is accepted. In this case, the proposal is immediately accepted if its score is 1 (indicating it is inside the region of interest) or the score is smaller than the score of the previous accepted point. Experiments are designed to test the various likelihood functions.

Algorithm 3.2: Adaptive Metropolis (adapted from [Vrugt \(2016\)](#))

Input: The likelihood function $L: \mathbb{R}^n \rightarrow \mathbb{R}$
The bounds of the uncertainty space
 i_{max} : The number of samples to be drawn
Output: A list of accepted samples and their respective likelihoods

- 1 $d \leftarrow$ number of dimensions
- 2 Generate empty arrays of length i_{max} for samples and scores
- 3 Create covariance matrix $cov \leftarrow 2.38^{\frac{2}{d}} \cdot Identity_matrix(d)$
- 4 $n_{accepted} \leftarrow 0$
- 5 $i \leftarrow 0$
- 6 **while** $i < i_{max}$ **do**
 - 7 $\delta \leftarrow$ an array of size $(d, 0)$, generated from a multivariate normal distribution with mean = 0 and covariance = cov
 - 8 Generate new proposal $p_{i+1} = p_i + \delta$
 - 9 Adapt and rescale new proposal to ensure it respects the bounds
 - 10 Calculate the likelihood $score_{i+1}$ of the proposal using the likelihood function L
 - 11 **if** $score_{i+1} \leq score_i$ **or** $score == 1$ **then**
 - 12 Accept proposal p_{i+1} :
 - 13 Add proposal to array of samples
 - 14 Add $score_{i+1}$ to list of scores
 - 15 $n_{accepted} += 1$
 - 16 **else**
 - 17 Reject proposal:
 - 18 Add previous sample p_i to array of samples
 - 19 Add previous $score_i$ to list of scores
- 20 Update array of chains to ensure all samples are within bounds
- 21 **return** $n_{accepted}$, $chains$, $scores$

3.2.3 Differential Evolution Markov Chain

Differential Evolution Markov chain (DE-MC) uses different chains to explore the posterior target distribution, to prevent premature convergence for complex problems. It combines the principles of the Metropolis selection rule with a genetic algorithm for population evolution ([Ter Braak, 2006](#)). Proposal samples are generated using differential evolution ([Storn and Price, 1997](#); [Price et al., 2005](#)):

$$\mathbf{X}_p^i = \gamma_d (\mathbf{X}^a - \mathbf{X}^b) + \zeta_d, \quad a \neq b \neq i \quad (3.1)$$

where:

- γ = scaling factor or jump rate; typically set to $2.38/2d$;

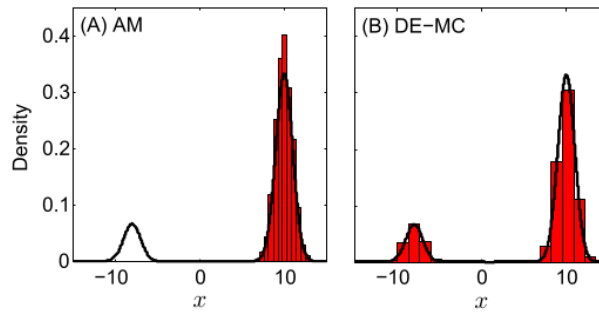


Figure 3.2: Histogram of the posterior distribution derived from (a) AM and (b) DE-MC. Reprinted from “Markov chain Monte Carlo simulation using the DREAM software package: Theory, concepts, and MATLAB implementation” by Vrugt, J.A., 2016, *Environmental Modeling and Software*, 75, 273–316.

- a and b are integer values from $[1, \dots, i-1, i+1, \dots, N]$ (N is the number of chains), i.e. two random other chains are selected, and the proposed sample is generated from the difference between these chains;
- ζ_d = a real number drawn from a normal distribution with mean 0 and standard deviation = 10^{-6} .

Additionally, DE-MC includes a way to enable transitions to different regions of interest, by setting the value of γ to 1 with a probability of 10% (Ter Braak, 2006). This functionality is demonstrated by figure 3.2.

Implementation used in experiments

Analogous to Adaptive Metropolis, the implementation of Differential-Evolution Markov Chain is based on Approximate Bayesian Computation, resulting in the acceptance condition as in line 13 of algorithm 3.3.

Apart from this, the following settings are used in the experiments for this thesis:

- Number of chains: 3 times the number of dimensions, as proposed by Ter Braak (2006)
- Crossover probability: 0.1, the default value, as proposed by Ter Braak (2006)

3.2.4 Differential Evolution Adaptive Metropolis

Compared to DE-MC, several additional features are implemented in DREAM to speed up convergence, specifically for non-linear, high-dimensional and multi-modal problems. In the following paragraphs, each component will be briefly explained, after which the algorithm settings used for the experiments in this thesis are listed.

Adaptive randomized subspace sampling

Each time a proposal sample is generated in DREAM, only a limited number of dimensions is updated. This is implemented in the calculation of the jump rate γ . An additional hyperparameter δ is the maximum number of chain pairs used to generate the jump to the proposal point. The default value of 3 results in one-third of the proposals being created with $\delta = 1$, another third with $\delta = 2$, and one third using $\delta = 3$ (Vrugt, 2016). Compared to DE-MC, the mode-jumping probability is higher (0.2 instead of 0.1), to enhance the exploration of various modes of the model outcome space.

By implementing subspace sampling, the required number of chains can be significantly reduced from $N \geq 2d$ (DE-MC) (Ter Braak, 2006) to $N < d$ (Vrugt, 2016).

Algorithm 3.3: Differential Evolution Markov Chain (adapted from [Vrugt \(2016\)](#))

Input: The likelihood function $L: \mathbb{R}^n \rightarrow \mathbb{R}$

The bounds of the uncertainty space

The number of chains

i_{max} : The number of samples to be drawn

p_{CR} : The crossover probability (=0.9)

Output: A list of accepted samples and their respective likelihoods

```

1  $d \leftarrow$  number of dimensions
2  $\gamma_{RWM} \leftarrow \frac{2.38}{\sqrt{2d}}$ 
3  $n_{accepted} \leftarrow 0$ 
4  $i \leftarrow 0$ 
5 while  $i < i_{max}$  do
6   Assign  $\gamma_{RWM}$  to  $g$  with probability  $p_{CR}$ , or 1 with probability  $(1 - p_{CR})$ 
7   for  $chain_i$  in  $range(chains)$  do
8     Select randomly (with equal probability) two other chains  $a$  and  $b$ 
9     Generate an array  $norm\_a$  of size  $d$  from a normal distribution with
      mean 0 and stdev 1.
10    Generate new proposal:  $p_{i+1} = p_i + g * (a - b) + 1e^{-6} * norm\_a$ 
11    Adapt and rescale proposal to ensure it respects the bounds
12    Calculate the likelihood  $score_{i+1}$  of the proposal using the likelihood
      function  $L$ 
13    if  $score_{i+1} \leq score_i$  or  $score == 1$  then
14      Accept proposal  $p_{i+1}$ :
15      Add proposal to chain
16      Add  $score_{i+1}$  to list of scores
17       $n_{accepted} += 1$ 
18    else
19      Reject proposal:
20      Add previous sample  $p_i$  to chain
21      Add previous  $score_i$  to list of scores
22 Update array of chains to ensure all samples are within bounds
23 return  $n_{accepted}, chains, scores$ 

```

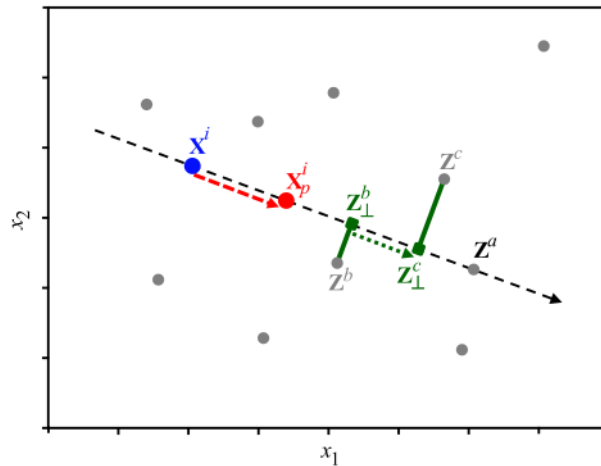


Figure 3.3: From [Vrugt \(2016\)](#): DREAM (ZS) algorithm: Explanation of the snooker update for a hypothetical two-dimensional problem using some external archive of $m = 10$ points (grey dots). Three points of this archive Z^a , Z^b and Z^c are sampled at random and define the jump of the i th chain, X^i (blue) as follows. The points Z^b and Z^c are projected orthogonally on to the dotted $X^i Z^a$ line. The jump is now defined as a multiple of the difference between the projections points Z_{\perp}^b and Z_{\perp}^c (green squares) and creates the proposal, X_p^i . The DREAM (ZS) algorithm uses a 90/10 % mix of parallel direction and snooker updates, respectively.

Outlier chain correction

One of the critical limitations of multi-chain MCMC methods is that chains may become ‘trapped’ in an unproductive region, because the size of the jumps is not large enough to escape this local optimum. Since new proposals are generated from the states of other chains, this slows (or even inhibits) convergence. The jumps of the ‘good’ chains are contaminated and the algorithm cannot converge (further explained in section 6.2.3) ([Gelman and Rubin, 1992](#); [ter Braak and Vrugt, 2008](#)).

The outlier chain detection implemented in DREAM uses the mean density (score) of the second half of each chain to assess the performance of each trajectory. Anomalous behavior is flagged, and outlier chains are moved to the position of one of the other chains. ([Vrugt et al., 2009](#)).

Snooker update

Additionally, the DREAM(ZS) algorithm implemented the *snooker update* to increase the diversity of the generated proposal samples.

Multi-Try

MT-DREAM(ZS) additionally implements multi-try sampling. This introduced an additional parameter, μ , which is the number of different proposals generated for each step for each chain. This additional feature works as follows ([Laloy and Vrugt, 2012](#)):

1. Generate μ trials $\mathbf{z}_1, \dots, \mathbf{z}_{\mu}$ from the current state of the chain, $j = 1, \dots, \mu$
2. Compute the score of each of the μ proposal points
3. Randomly select one candidate point \mathbf{z}_j from step 1, with probability of each trial proportional to its score
4. Generate reference points $\mathbf{x}_1^*, \dots, \mathbf{x}_{\mu-1}^*$ from the current state of the chain
5. Set \mathbf{x}_{μ}^* to \mathbf{x}_{t-1}

6. Accept \mathbf{z} with probability:

$$p(\mathbf{x}_{t-1}, \mathbf{z}) = \min \left\{ 1, \frac{\pi(\mathbf{z}_1) + \dots + \pi(\mathbf{z}_k)}{\pi(\mathbf{x}_1^*) + \dots + \pi(\mathbf{x}_k^*)} \right\} \quad (3.2)$$

where $\pi(\mathbf{z}_i)$ and $\pi(\mathbf{x}_i^*)$ are the performance scores of the candidate points and reference points, respectively.

The trial proposals can all be evaluated in parallel. The default value of $\mu = 5$.

Implementation used in experiments

Again, the implementation of Differential-Evolution Adaptive Metropolis is based on Approximate Bayesian Computation, resulting in the acceptance condition as in line 15 of algorithm 3.4.

Not all of the features previously described are used in the experiments for this thesis. The following settings are used:

- Number of chains: 3, following the recommendation by [Vrugt \(2016\)](#).
- Prior distribution: uniform, so an uninformed prior is used.
- Multi-try: False. The logic implemented by multi-try, where the acceptance probability is based on the ratio of the sums of scores, is incompatible with the principles of ABC, where the proposal sample is accepted if it outperforms the current point.
- Random start: True, to evaluate how DREAM performs when the user has no prior knowledge. Since the starting position has a relatively large impact on the performance of the algorithm when using a relatively low number of function evaluations, replications will be performed of each experiment.

3.2.5 Shape of the likelihood function

The algorithms described above use a deterministic acceptance condition: proposals are always accepted if they are within the region of interest, or if they perform better than the previous point. This performance is defined in the *likelihood function*, which is user-defined. The likelihood function takes the parameter vector generated by the algorithm, calls the model (e.g. the barrel functions), evaluates the outcomes of interest, and returns the ‘likelihood’ of the parameter vector. This logic is presented as pseudocode in 3.5. The likelihood functions are designed to have a uniform, highest value within the region of interest, and decrease with increasing distance.

Since the likelihood function plays a central role in the performance of the algorithms, various shapes were examined to find the function that led to the best performance in each case. The examined shapes are inspired by the activation functions in neural networks, which are used to map outcome values to the extent of activation of the hidden neuron ([Goodfellow et al., 2016](#)). This is similar to the logic of the likelihood function: the outcome values (which could be anything from $-\infty$ to ∞), should be translated to ‘within region of interest’ or ‘outside region of interest’.

The likelihood shapes presented in figure 3.4 are used in combination with the adapted implementations of Adaptive Metropolis, Differential-Evolution Markov Chain and DREAM(ABC) (as presented in section 5.1.2 for each of the test problems).

The ‘distance’ likelihood function (figure 3.4a) decreases linearly with increasing distance. Therefore, any step closer to the region of interest is immediately accepted, and the probability of making steps backwards is equal regardless of the position with respect to the barrel. The sigmoidal-shaped hyperbolic tangent functions are more lenient close to the region of interest, decrease quickly with increasing distance,

Algorithm 3.4: Differential Evolution Adaptive Metropolis (adapted from [Vrugt \(2016\)](#))

Input: The likelihood function $L: \mathbb{R}^n \rightarrow \mathbb{R}$
The number of chains (N)
A prior distribution i_{max} : The number of samples to be drawn
 p_{CR} : The crossover probability (=0.8)
Output: A list of accepted samples and their respective likelihoods

```

1  $i \leftarrow 0$ 
2 while  $i < i_{max}$  do
3   for  $chain_i$  in  $range(chains)$  do
4     Select randomly  $r_1, \dots, r_\delta$  from  $[1, \dots, chain - 1, chain + 1, \dots, N]$  Set  $\mathbf{b}$  to
       be an empty set
5     for  $m = \{1, \dots, N\}$  do
6       Draw a value  $z_e$  from a standard uniform distribution
7       if  $z_e \geq (1 - p_{CR})$  then
8         | Add dimension  $m$  to the set  $\mathbf{b}$ 
9        $p^* \leftarrow$  number of elements of  $\mathbf{b}$ 
10      Set  $\gamma$  to  $2.4 / \sqrt{2\delta p^*}$  with 80% probability, or 1 with 20% probability
        Generate an array  $norm\_a$  of size  $d$  from a normal distribution with
        mean 0 and stdev 1.
11      Generate proposal of the  $j$ th chain:
        
$$A_{(i)(P)}^j = A_{(i)}^j + \gamma_{(\delta, d^*)} \left( \sum_{l=1}^{\delta} A_{(r_l), \mathbf{b}}^j - \sum_{l=\delta+1}^{2\delta} A_{(r_l), \mathbf{b}}^j \right) + 1e^{-6} * norm\_a$$

12      Adapt and rescale proposal to ensure it respects the bounds
13      Calculate the likelihood  $score_{i+1}$  of the proposal using the likelihood
        function  $L$ 
14    for  $chain_i$  in  $range(chains)$  do
15      if  $score_{i+1} \leq score_i$  or  $score == 1$  then
16        | Accept proposal  $p_{i+1}$ :
17        | Add proposal to chain
18        | Add  $score_{i+1}$  to list of scores
19        |  $n_{accepted} += 1$ 
20      else
21        | Reject proposal:
22        | Add previous sample  $p_i$  to chain
23        | Add previous  $score_i$  to list of scores
24  Detect and reset outlier chains by comparing the mean log-densities of the
    second half of each of the  $N$  chains
25 return  $n_{accepted}, chains, scores$ 

```

Algorithm 3.5: Likelihood function - generic

Input: Parameter vector (sample)
Model function
Threshold that defines the region of interest
Output: Likelihood of the sampled parameters

```

1 model_outcome = model(parameter_vector)
2 if  $model\_outcome < threshold$  then
3   | likelihood = 1
4 else
5   | likelihood = decreasing with increasing distance (figures 3.4)
6 return likelihood

```

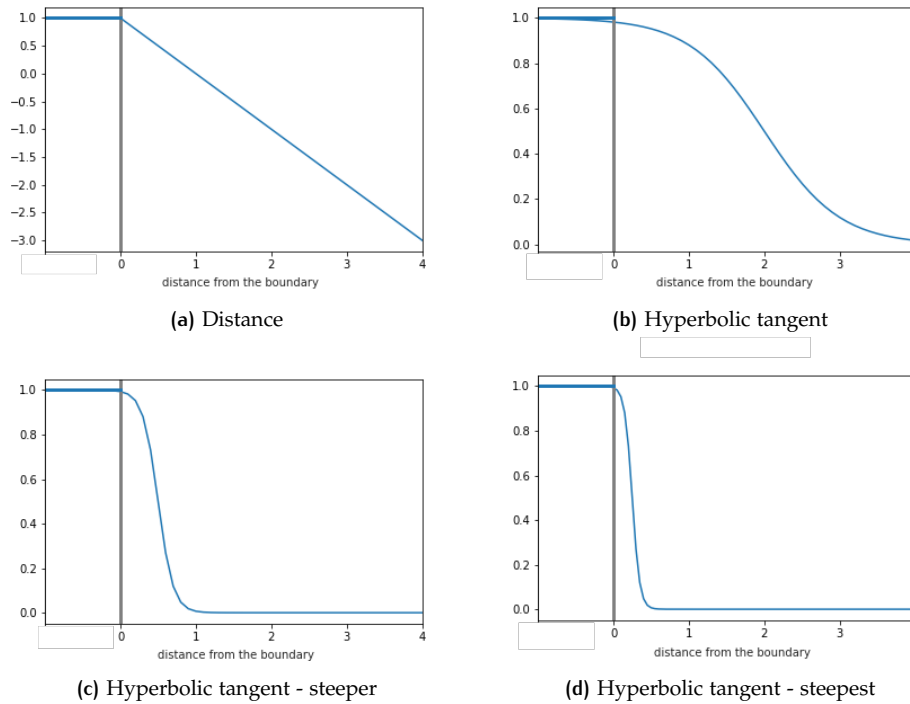


Figure 3.4: Graphs of the examined likelihood functions. The grey vertical line depicts the boundary of the region of interest. The returned value of the likelihood function (y-axis) decreases with increasing distance from this boundary.

and flatten afterwards. Therefore, if the chain is far away from the region of interest, it doesn't necessarily move closer. In large spaces, with relatively small regions of interest this may be inefficient. The probability of moving further away from the region of interest is very small when the chain is close, due to the steep slope. A steeper slope emphasizes this behavior.

Following the conclusions drawn from the experiments discussed in appendix A, the 'distance' likelihood function will be used for each algorithm for the experiments conducted for this thesis.

3.3 PERFORMANCE MEASURES

A set of performance measures is selected to describe the suitability of each approach for Scenario Discovery, for a variety of test problems. The goal of Scenario Discovery is to find all combinations of ranges of parameter values leading to a behavior of interest and communicate these in an intuitive way. In other words, the objective is twofold: the approach should be suitable to collect a representative sample of the regions of interest in order to analyse it, and the collected sample should be readily processed to inform decision-makers.

These goals can be divided into three categories. Firstly, the results of the analysis should be *easily interpretable*. Secondly, the *quality* of the sample should be high, i.e. suitable for a detailed analysis. Thirdly, the collection of the sample should be *efficient*, to facilitate the analysis of complex problems.

In the following subsections, various performance measures from literature are discussed for each of these categories. In the final subsection, a set of performance measures is presented that, together, provide a comprehensive understanding of the relative performance of the various algorithms and corresponding approaches.

3.3.1 Interpretability

Interpretability is one of the performance measure proposed by [Lempert et al. \(2008\)](#). It refers to the ease with which decision-makers can understand the results. In the context of Scenario Discovery using PRIM or CART, it is often implemented by restricting the number of boxes to three or four, with each a maximum of two to three restricted dimensions.

In this thesis, interpretability more broadly refers to the ease with which decision-makers can understand the results. Instead of trying to quantify this measure, it will be qualitatively discussed by comparing the results of Scenario Discovery using an independent sampling approach to the results of post-processing the MCMC samples. Chapter 4 will address the differences between independent and dependent sampling with regards to interpretability in more detail.

3.3.2 Quality

To use MCMC for Scenario Discovery, it is imperative that the resulting sample can be adequately analysed, yielding ranges of parameter values and/or intuitive visualizations, which can be readily interpreted by decision-makers.

Correct representation of the regions of interest

Firstly, the sample should be validated, by checking whether the correct regions of interest are found. This entails two checks: 1) are all the regions of interest found, and 2) are the ranges of parameter values roughly correct. These checks can easily be performed on the known test problems presented in this thesis. Findings may be extended to the performance on unknown problems, where these checks are impossible.

Fraction of points of interest

The fraction of points of interest indicates how many of the samples in the chain (given a certain burn-in period) are cases of interest. For MCMC algorithms, this indicates how well the chain characterizes the posterior distribution. Figure 3.5 illustrates how a different shape of the likelihood function of an MCMC algorithm may affect the fraction of interest, and, therefore, how well the posterior can be characterized.

This performance measure can also be used to compare the computational efficiency between MCMC algorithms and Latin Hypercube Sampling. It exposes the limitation of LHS for the analysis of high-dimensional problems, especially when the regions of interest are small and/or sparse.

3.3.3 Efficiency

Initial exploration of the application of the R-statistic for convergence yields poor results for the test shapes, where \hat{R} immediately dives under 1.2, and stays there. Potentially this is because the chains have not sufficiently mixed, as the Gelman Rubin diagnostic can only be used after a burn-in period. However, calculations at any nfe yield similar results. Therefore, it is not used for now.

The fraction of interest and acceptance rate together give a good idea of the computational efficiency and performance of the MCMC algorithms. It is important to consider them together: an exceptionally high acceptance rate may also indicate a sloppy sample, and an exceptionally high fraction of interest could also mean many of the samples in the chain are in fact the same point. In this case, the algorithm found a region of interest, but never moved on, because any proposed sample is worse than the current position.

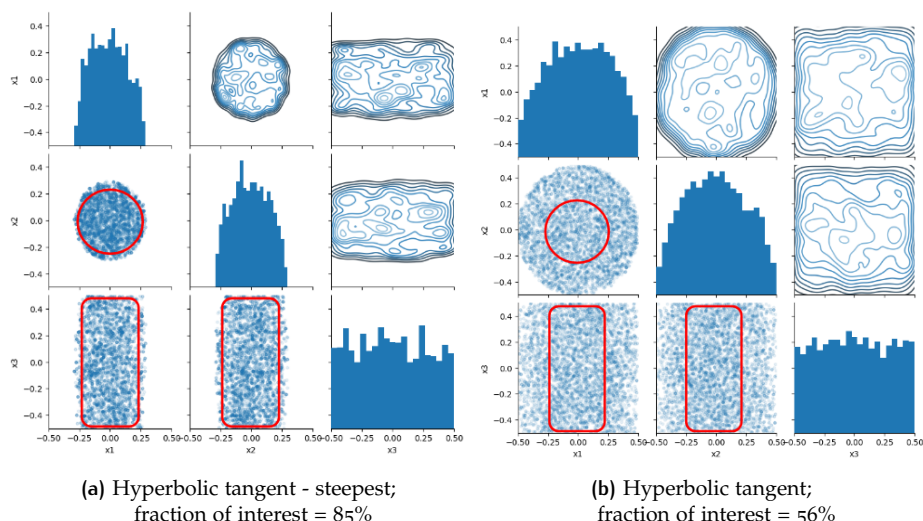


Figure 3.5: Pairplots of the Adaptive Metropolis algorithm (using the Metropolis Ratio for acceptance/rejection) applied to the ‘upright barrel shape’ with nfe=10,000, for various likelihood functions. The region of interest is indicated in red.

Table 3.3: Input to various algorithms to guarantee the same number of function evaluations

	Number of chains	Function evaluations			
		1000	3000	15000	30000
LHS	n.a.	1000	3000	15000	30000
AM	1	1000	3000	15000	30000
DE-MC	9	111	333	1111	3333
DREAM(ABC)	3	335	1000	5000	10000

Number of function evaluations

Sadegh and Vrugt (2014) compare DREAM(ABC) to Rejection Sampling (ABC) and to Population Monte Carlo (ABC) by comparing the number of function evaluations (nfe) required to find 1,000 solutions of interest. They record the nfe and the acceptance rate to compare the computational efficiency of the various algorithms.

Laloy and Vrugt (2012) compare the computational efficiency of DREAM(ZS) and MT-DREAM(ZS). Here, the number of Computational Time Units (CTU) is recorded. For sequential methods, this is equivalent to using the nfe. However, for DREAM(ZS), $CTU = FE/N$, where N is the number of chains. For MT-DREAM(ZS), $CTU = FE/(N * (k - \frac{1}{2}))$, where k is the number of tries per iteration. However, Chu et al. (2014) criticize the use of CTU as a measure of computational efficiency, as the comparison between two algorithms is only fair if both are given the same number of threads.

Therefore, the number of function evaluations will be varied for each algorithm to assess whether this affects the performance of each sampling approach differently. Since DE-MC and DREAM are multi-chain methods, the ‘number of iterations’ specified for these algorithms will deviate from the number of function evaluations, since the chains are merged in the end. Table 3.3 specifies these numbers. This implies that the multi-chain methods have a lower number of iterations *per chain*, compared to the single-chain method AM. However, multi-chain methods are designed to be faster than running the same number of function evaluations sequentially in a single chain, as multiple chains are ran in parallel.

MCMC methods typically use a burn-in period: the first number of samples are discarded. The analyses in this research use a burn-in period of 10%, or 100 iterations if 10% is lower than that (unless otherwise indicated).

Acceptance rate

The acceptance rate is the percentage of proposal points which is accepted, rather than rejected. This statistic is used to measure the relative efficiencies of DREAM(ABC) (Sadegh and Vrugt, 2014), DREAM(ZS) (Vrugt, 2016), and MT-DREAM(ZS) (Laloy and Vrugt, 2012). A higher acceptance rate indicates a higher efficiency. However, if the acceptance rate is too high, the accuracy of the sample may be low (Laloy and Vrugt, 2012).

3.3.4 Selection of measures for research

To summarize, for each of the broad requirements a set of performance measures is selected that, together, provide insight in the performance of the approach for Scenario Discovery.

1. Interpretability
 - A qualitative discussion of the relative interpretability of the approaches
2. Quality
 - Correct representation (check)
 - Fraction of cases of interest
3. Efficiency
 - Acceptance rate
 - Number of function evaluations

The next chapter will address the differences between independent and dependent sampling with regards to interpretability. The ease of interpretation is related to the processing and presentation of the results and is therefore equal for each MCMC algorithm.

The subsequent chapter evaluates a number of MCMC algorithms on the basis of the remaining performance measures.

4 | VISUALIZATION AND COMMUNICATION OF RESULTS

This chapter explores the potential of dependent sampling approaches for Scenario Discovery by proposing a method for the processing of the results, yielding outcomes that are relevant in a policy-making context. These outcomes and the processing steps are compared to the original independent sampling approach. Throughout the chapter, the 50d, small test problem with two disjoint barrels in separate dimensions is used for demonstration of the approaches.

4.1 INDEPENDENT SAMPLING

The methodology for Scenario Discovery (as described in section 2.1) includes processing of the results using PRIM, which highlights the dimensions causing the behavior of interest. This greatly improves the interpretability of the results, by immediately presenting decision-makers with ranges of input parameter values causing the behavior of interest.

As a baseline for the interpretability of the results of MCMC approaches, PRIM is applied to the small, 50d test case. This test problem contains a sufficient sample of the cases of interest because of the small interval, but cannot be intuitively interpreted from a scatterplot due to its high number of dimensions. The same analysis is performed for each test case discussed in this chapter.

As demonstrated in figure 4.1, it is possible to identify the ranges of input parameter values that define the region of interest using two boxes. The first box describes the barrel defined by x_1, x_2, x_3 , and the second box describes the barrel defined by x_{47}, x_{48}, x_{49} . PRIM ignores the remaining 44 empty dimensions. Similar to the 3d plots in figure 3.1, the first barrel cannot be visually distinguished because of the cases of interest from the second barrel. The ranges of values for uncertain parameters defining the regions of interest are presented in table 4.1, as would be presented to decision-makers.

It is important to realise that this analysis cannot be performed for the test problems with a larger parameter space, since even the sample with 30,000 cases did not provide a sufficient characterization of the input space (see section 5.2.2).

4.1.1 Applicability of original approach to MCMC sample

The definitions of density and coverage from Lempert et al. (2008) are not directly applicable to dependent samples. The reasoning for this is outlined below, along with their potential implementations for dependent samples.

Table 4.1: Ranges of values for uncertain parameters defining the regions of interest, for LHS sample of 50d, small test problem using PRIM.

	box 1			box 2	
	min	max		min	max
x_1	-0.131531	0.132578	x_{47}	-0.413606	0.394481
x_2	0.108813	0.397337	x_{48}	-0.367195	-0.118585
x_3	-0.401516	0.445789	x_{49}	-0.129931	0.146916

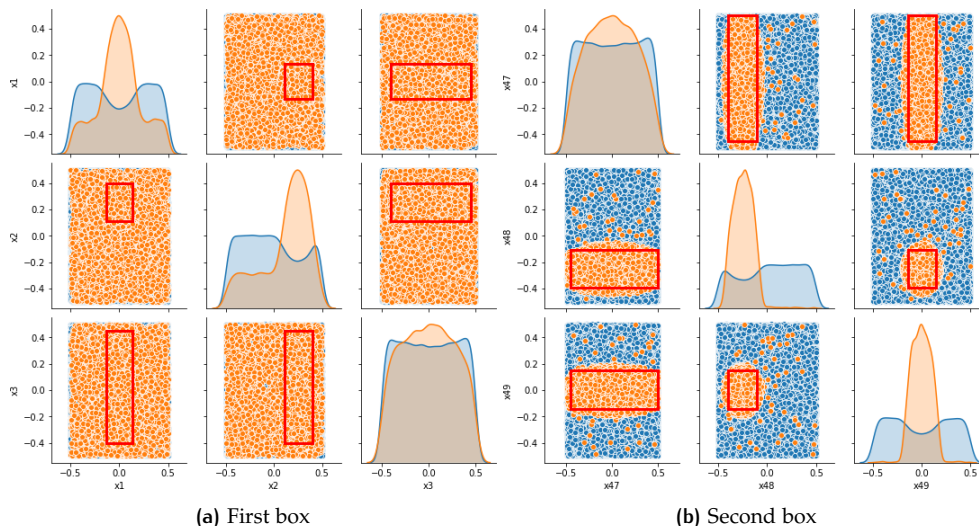


Figure 4.1: PRIM performed on LHS (independent sampling) results for the 50d, small test problem. Orange dots are cases of interest, blue dots are not.

Coverage and density, the two quantifiable performance measures proposed by Lempert et al. (2008), are not directly applicable to a Markov chain. However, they may be adapted to provide a similar insight for MCMC results as they do for LHS results.

Coverage

Coverage was originally defined as the number of cases of interest within the box, divided by the total number of cases of interest (Lempert et al., 2006). Since the MCMC-ABC sample only consists of cases of interest, coverage cannot be calculated directly. Since its probability distribution reflects the true posterior, the percentage of the integral of the probability distribution within the box is analogous to coverage: as the size of a box decreases, the portion of the distribution within the box decreases.

Density

Latin Hypercube produces a sample which is approximately uniform for each parameter, containing both cases of interest and cases *not* of interest, which allows for the calculation of the density. Density was originally defined as the number of cases of interest within the box, divided by the total number of cases within the box (Lempert et al., 2006). Instead, dependent sampling methods *reject* proposed parameter values that do not lead to behavior of interest, yielding a chain with a very high fraction of cases of interest. Thus, the density of a DREAM sample cannot readily be determined.

Potentially, one could also record these rejected proposals, and use these data points as the ‘cases not of interest’ to calculate the density. However, these points would be mostly concentrated at the boundaries of the regions of interest, rather than cover the space. Because even if they are recorded, they are not accepted, and therefore new points are being generated from the position of the last *accepted point*, which is within the region of interest. The distribution of rejected proposals will be skewed due to the limited jump rate.

Another option is to use MCMC sampling to derive the relevant dimensions using the Kolmogorov-Smirnov test and take a LHS sample of this subset of dimensions. Or, one could go one step further and extract coarse ranges of the parameter values to use as inputs for Latin Hypercube. This sample of a subspace of the uncertainty space could be further analysed using PRIM or CART.

The remainder of this chapter will explore the possibility of directly using the MCMC sample for further analysis, without the need of additional sampling.

4.2 DEPENDENT SAMPLING

Similar to what PRIM does for the independent sample, the results of the MCMC algorithms should be post-processed to present the ranges of values for uncertain parameters defining the regions of interest. To this the end, the following features should be derived:

1. the relevant dimensions to which the behavior of interest is sensitive
2. the combinations of ranges of parameter values leading to behavior of interest. I.e. for this test problem, a combination of x_1 , x_2 , x_3 may lead to behavior of interest, regardless of the values for x_{47} , x_{48} and x_{49} , and vice versa.

Firstly, the samples should be ‘filtered’ to derive the dimensions of interest. A property which could be used to derive the dimensions causing the behavior of interest is the fact that the input space is non-uniformly sampled, i.e. regions of interest are sampled significantly more often: the sampling ‘concentrates’ in the regions of interest. After the algorithm converges, the sample represents the target distribution (similar to the orange density function in figure 4.1b). When comparing subfigures 4.3a and 4.3a, the difference is evident: sampling for x_2 (one of the dimensions of interest) is clearly not uniform. An often-used method to quantify the different shapes of these distributions is the Kolmogorov–Smirnov test (KS test) (Massey, 1951).

Secondly, a potential approach is explored to extract the ranges from these dimensions. The probability density of the parameter values is estimated from the sample and the part within that sample with high density is derived.

By slicing the dataframe at the previously found ranges of parameter values, recalculating the KS statistic for this subset and finding the corresponding ranges of parameter values for the other dimensions, the combinations of ranges of parameter values leading to behavior of interest are found.

This approach is demonstrated for the 50d, small problem, which shows how multiple dimensions in distinct subspaces are found. Finally, three test cases are discussed to demonstrate the possibility to distinguish multiple regions of interest within a dimension.

4.2.1 Deriving the dimensions of interest

The Kolmogorov–Smirnov test compares a sample to a probability distribution by calculating the maximum distance between the sample’s cumulative density function (CDF) and the CDF of the reference distribution (as illustrated in figure 4.2). If this statistic is low, the sample follows the reference distribution. If it is higher, the hypothesis of resembling the reference distribution is rejected.

This test can be used to distinguish dimensions in which the sampling is *not* uniform, indicating a region of interest. This is illustrated in figures 4.3 and 4.4, where the KS statistic is plotted for each dimension. The value is calculated by removing the burn-in period and merging the chains.

Since this is a stylized problem, where the regions of interest are known beforehand, the results can be validated. The dimensions x_1 , x_2 , x_{48} and x_{49} have a much higher value for the KS statistic, as demonstrated in figure 4.10. Since the barrels occupy almost the entire space in dimensions x_3 and x_{47} , most PRIM boxes also do not include these dimensions.

Evidently, the Kolmogorov–Smirnov test, using a uniform distribution with $loc=-0.5$ and $scale=1$ (i.e. bounds of $[-0.5, 0.5]$) point to the dimensions that contain

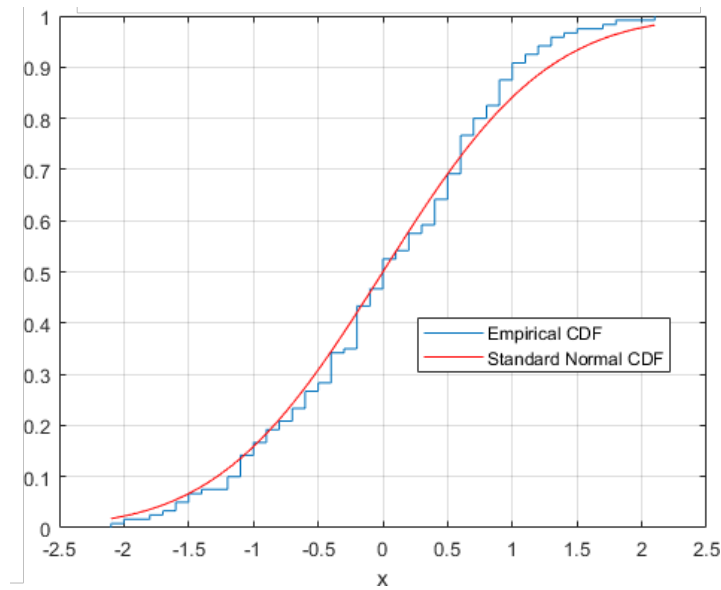
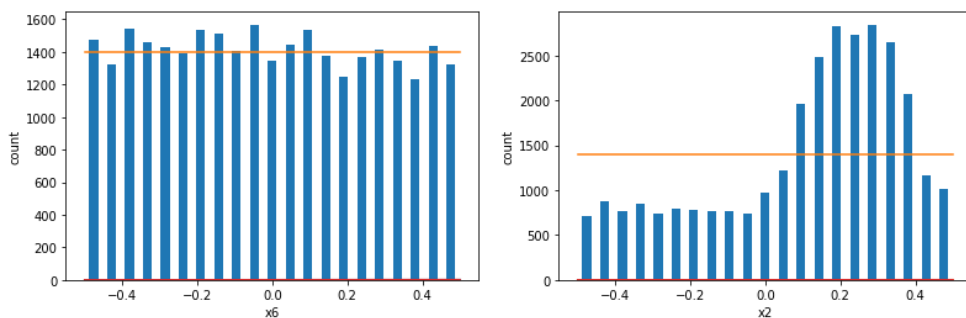


Figure 4.2: Illustration of the Kolmogorov–Smirnov test for a normal distribution, with the cumulative density function of the sample in blue and the reference distribution in red. The pairwise distances between the sample and the reference distribution are calculated at each point. The maximum of these distances is the recorded statistic. Adapted from Mathworks (2020).



(a) Dimension x_6 , which does not contribute to the region of interest. KS statistic = 0.02. (b) Dimension x_2 , which contributes to the region of interest. KS statistic = 0.23.

Figure 4.3: Histograms of the distribution of samples for two dimensions: one which does and one that does not contribute to the region of interest; to illustrate how the difference may be distinguished from an MCMC sample. The orange line is a uniform distribution.

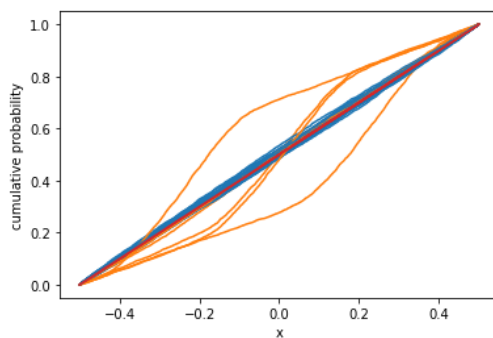


Figure 4.4: Cumulative probability functions of x_1 to x_{50} in blue. Dimensions with a KS statistic of > 0.1 (i.e. x_1, x_2, x_{48} and x_{49}) are coloured orange for clarity. The cumulative probability of a uniform distribution is displayed in red.

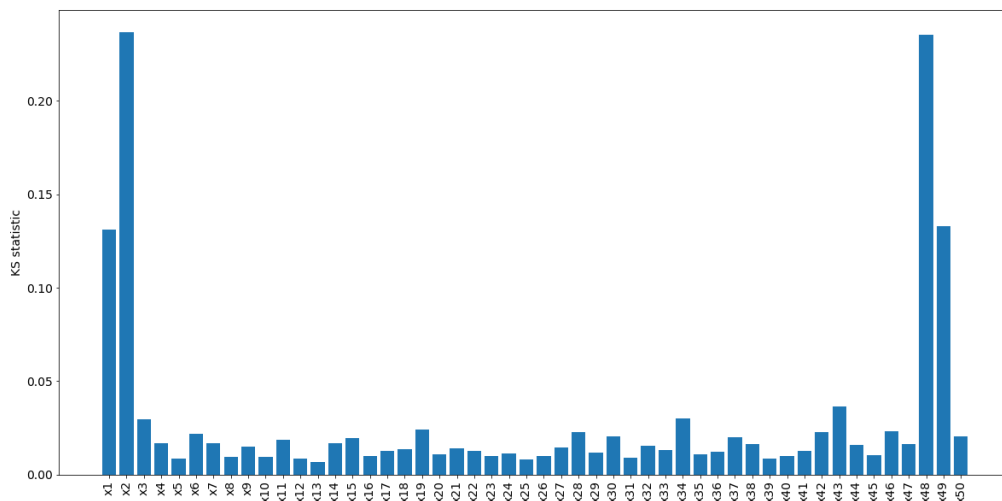


Figure 4.5: Barplot of the value of the KS statistic for each of the dimensions in the DREAM(ABC) sample of the 50d, small test problem with $n_{fe}=30,000$. The KS statistic is calculated by comparing to a uniform distribution with bounds $[-0.5, 0.5]$.

the region of interest. It is therefore an applicable and useful preprocessing step to visualize the results of Scenario Discovery using a dependent sampling approach. For general applicability, it will be useful to define a threshold below which the dimension will not be considered. In this case, a threshold of ≈ 0.05 yields the desired results, which corresponds to the threshold value recommended in literature (Massey, 1951).

Following Sadegh and Vrugt (2014) in their journal paper presenting DREAM(ABC), the selected dimensions of the DREAM(ABC) sample ($x_1, x_2, x_3, x_{47}, x_{48}$ and x_{49}) are visualized in a pairplot, showing the pairwise relationships in the sample. This plot (figure 4.6) gives insight in the ranges of parameter values of interest. A trained eye may also notice that x_1, x_2 , and x_3 , and x_{47}, x_{48} and x_{49} form two distinct regions of interest. However, the interpretation is not intuitive, and therefore not suitable for communication to decision-makers.

Contribution to the performance measures

The KS-test yields additional information, which the previous performance measures ‘acceptance rate’ and ‘fraction of cases of interest’ did not yet reveal. Firstly, it allows one to check whether *all* regions of interest are found, or whether the algorithm gets stuck in one region of interest and fails to explore the entire uncertainty space.

Secondly, it allows to draw more quantitative conclusions as to which number of function evaluations is sufficient to characterize the regions of interest. Simply plotting the samples already suggest that the regions of interest cannot be readily distinguished, but this is highly qualitative. The histograms of the KS-statistic (figure 5.11b) quantifies and supports this hypothesis, since most irrelevant dimensions do not satisfy the threshold of 0.05.

4.2.2 Regions of interest

When the dimensions causing the region of interest are determined, the specific combinations of ranges of parameter values describing the region(s) of interest can be derived.

Ranges of input parameter values leading to an outcome of interest

Besides visual inspection, the ranges of input parameter values leading to an outcome of interest may be determined by visual inspection of the kernel density estimation

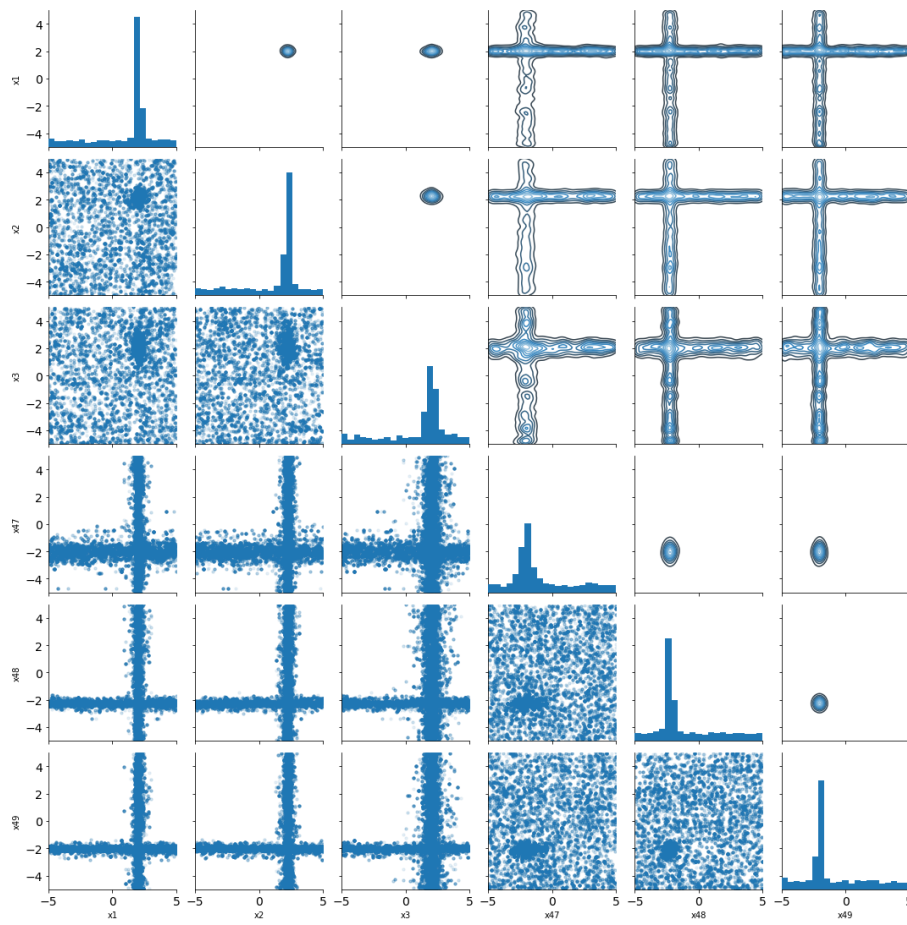


Figure 4.6: Pairplot of the selected dimensions of the DREAM(ABC) sample of the 50d, large test problem.

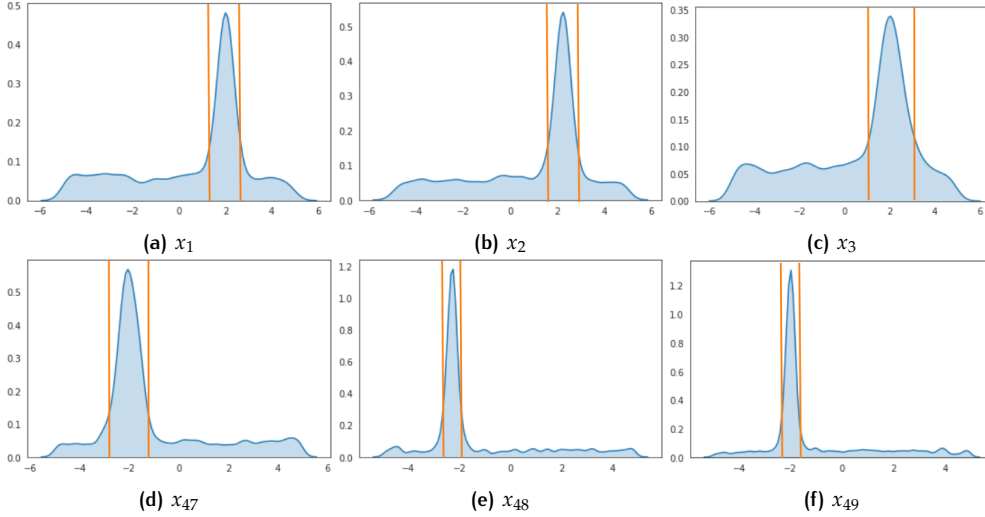


Figure 4.7: Probability density for each dimension of interest for the large, 50d test problem, generated by kernel density estimation. The ranges of parameter values are indicated by the vertical orange lines.

(kde) of each dimension or the raw data. KDE is a machine learning technique for data smoothing of finite samples (Rosenblatt, 1956; Parzen, 1962). It transforms the discrete sample into a continuous distribution of the sampling frequency.

Figure 4.7 shows the KDE plots of the selected dimensions. Vertical lines indicating the boundaries of the region of interest are derived using a threshold of 0.1 for the probability density. The parameter values with a density that exceeds this threshold are within a region of interest. This empirically derived threshold value of 0.1 yields results that are in agreement with the PRIM results, but may be varied to include more or fewer sampled points.

The bounds for the small test problem are also presented in table 4.2 and contrasted with the values found by PRIM. While the bounds found using this method vary from the values found by PRIM, the KDE plots of the MCMC samples resembles the KDE of the cases of interest, as visualised in figure 4.1. It is important to realise that both the box choice in PRIM and the choice of a threshold value here are highly subjective and largely determine the similarity of the values.

Table 4.2: Ranges of parameter values delineating the region of interest for the 50d, small test problem

	LHS		DREAM(ABC)	
	min	max	min	max
x_1	-0.16	0.17	-0.15	0.16
x_2	0.08	0.41	0.08	0.40
x_{48}	-0.38	0.10	-0.41	0.09
x_{49}	-0.13	0.14	-0.15	0.16

Combinations of parameter values

By iterating over the dimensions of interest, the combinations of parameter values leading to behavior of interest; i.e. the distinct regions of interest can be found. The process is as follows:

1. Select the next dimension of interest

2. Slice the dataframe so it only contains samples within the range derived from the KDE plot in the previous step of this dimension of interest and recalculate the KS statistic for the all other dimensions.
3. Select the dimension with the highest value for the KS statistic
4. Determine the range of parameter values leading to behavior of interest for this dimension
5. Repeat steps 2-4 until the KS statistics are low for each dimension; i.e. no other dimension contributes to the region of interest
6. Repeat for each dimension of interest found, until all are accounted for

4.2.3 Demonstration for three cases

This process can be explained and demonstrated using three examples: a case with one region of interest, and two case with two regions of interest: one where the regions of interest are in separate dimensions, and one where they are in the same dimensions. The following paragraphs will go through the steps to illustrate.

One region of interest

The test problem used in this paragraph is the upright barrel in a 3d, small uncertainty space. The KS test yields high values for x_1 and x_2 , so the steps are repeated for these two dimensions. Again, the barrel shape extends to the borders of the third dimension, so this dimension is only included if the threshold value is significantly increased. This is consistent with the results of [Lempert et al. \(2008\)](#).

1. First, select x_1 . The region of interest lies between -0.20 and 0.18.

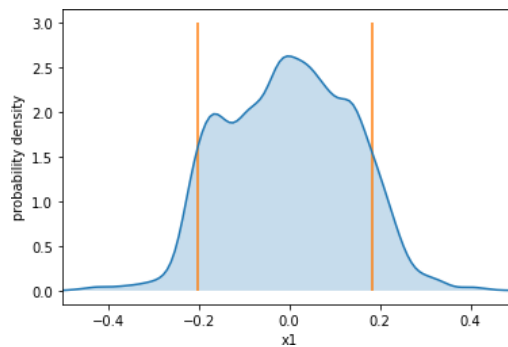


Figure 4.8: Dimension of interest x_1

2. Recalculating the KS statistic given $-0.20 < x_1 < 0.18$ yields a value of 0.24 for x_2 and 0.03 for x_3 .
3. x_2 is selected.
4. The area between -0.2 and 0.19 exceeds the threshold and therefore delineates the region of interest.
5. Recalculating the KS statistic for x_3 , given $-0.20 < x_1 < 0.18$ and $-0.20 < x_2 < 0.19$ yields a KS score of 0.03, which is not significant.
6. Repeating these steps for x_2 yields the same region of interest, so there is only one region of interest, defined by $(-0.20 < x_1 < 0.18$ and $-0.20 < x_2 < 0.19)$.

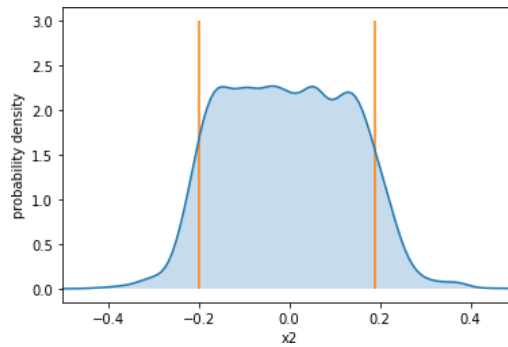


Figure 4.9: Dimension of interest x_2 , given $-0.20 < x_1 < 0.18$

The steps described above can be summarized in a table of KS values, given the ranges derived in each step. The conditions presented in the column names are cumulative, i.e. for the last column all previous conditions are also imposed. This manner of reporting will be used for the remainder of the cases.

Table 4.3: KS statistic for x_1, x_2, x_3 , given the conditions found in each step.

	none	$-0.20 < x_1 < 0.18$	$0.13 < x_2 < 0.38$
x_1	0.25	-	-
x_2	0.25	0.24	-
x_3	0.04	0.03	0.03

The values for the boundaries of the regions of interest are consistent with the results of independent sampling combined with PRIM.

Two regions of interest in the same dimensions

The test problem used in this paragraph is the disjoint barrels in a 3d, small uncertainty space. This problem is interesting, because PRIM finds two regions of interest, which partially occupy the same dimensions: one region of interest is defined by x_1 and x_2 , and the other by x_2 and x_3 . The steps proposed in sections 4.2.1 and 4.2.2 are followed to find these regions of interest from the DREAM(ABC) sample.

Table 4.4 presents the KS statistic for each dimension for each step. The leftmost column shows the KS statistic for each dimension without restrictions. The range of parameter values driving the behavior is derived for the first dimension and imposed as a rule. Repeating this process with this subset of the sample yields the first region of interest, defined by $-0.13 < x_1 < 0.13$ and $0.13 < x_2 < 0.38$.

Table 4.4: First round of the process: KS statistic for x_1, x_2, x_3 , given the conditions found in each step.

	none	$-0.13 < x_1 < 0.13$	$0.13 < x_2 < 0.38$
x_1	0.17	-	-
x_2	0.08	0.29	-
x_3	0.19	0.12	0.06

For the full sample, the KS statistic of x_1 and x_3 both exceed the threshold. Therefore, the process of finding the regions of interest will be repeated for x_3 .

These can be validated by comparison to the results of a PRIM analysis of a Latin Hypercube sample. Table 4.6 presents the regions of interest found by LHS/PRIM and using this method on a DREAM(ABC) sample.

The values describing the regions of interest are similar. The differences between the regions derived from the dependent sampling approach and the LHS/PRIM results are due to the box choice in PRIM (i.e. the specific trade-off between density

Table 4.5: Second round of the process: KS statistic for x_1, x_2, x_3 , given the conditions found in each step.

	none	$-0.20 < x_3 < 0.20$	$-0.42 < x_2 < -0.09$
x_1	0.17	0.12	0.05
x_2	0.08	0.24	-
x_3	0.19	-	-

Table 4.6: Regions of interest found by a PRIM analysis of a Latin Hypercube sample, contrasted with the results of the proposed methodology for dependent sampling.

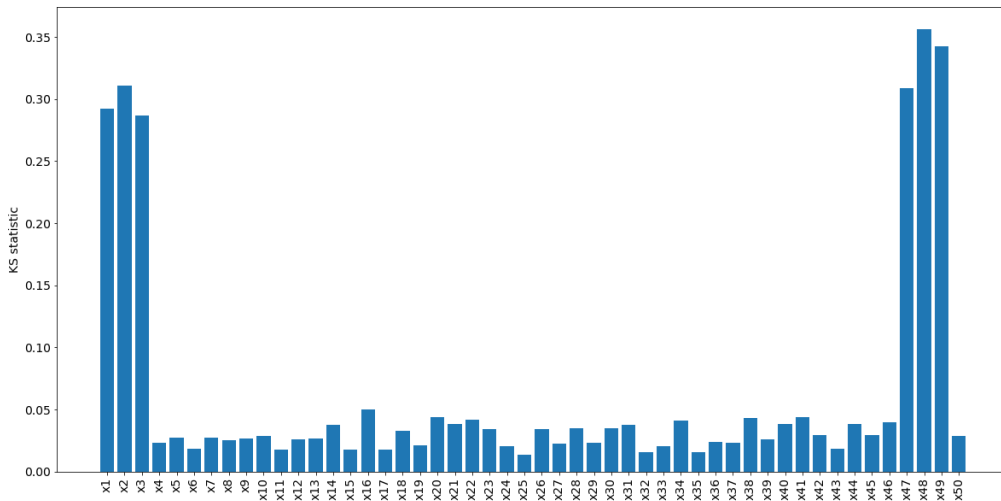
	First region				Second region			
	LHS		DREAM		LHS		DREAM	
	min	max	min	max	min	max	min	max
x_1	-0.16	0.17	-0.13	0.13	-	-	-	-
x_2	0.08	0.41	0.13	0.38	-0.38	-0.10	-0.42	-0.09
x_3	-	-	-	-	-0.13	0.14	-0.20	0.20

and coverage that is chosen) and the choice of the threshold value for the dependent sampling approach.

Two regions of interest in separate dimensions

The test problem used in this paragraph is the disjoint barrels in a 50d, large uncertainty space. The regions of interest in this test problem are placed in *different* dimensions.

Performing the KS test for the 50d, large test problem proves that, when the region of interest is much smaller than the uncertainty space, the results of the KS test are in line with the theory. Indeed, the dimensions of interest are $x_1, x_2, x_3, x_{47}, x_{48}$ and x_{49} , which have a much higher value for the KS statistic, as demonstrated in figure 4.10. So the steps proposed in section 4.2.2 will be repeated for each of these six dimensions.

**Figure 4.10:** Barplot of the value of the KS statistic for each of the dimensions in the DREAM(ABC) sample of the 50d, large test problem with $n_{fe}=30,000$. The KS statistic is calculated by comparing to a uniform distribution with bounds $[-5, 5]$.

Starting with x_1 , the first region of interest is found, defined by x_1, x_2 and x_3 .

The process is repeated beginning with x_2, x_3, x_{47}, x_{48} and x_{49} . The first two yield the same region of interest as x_1 . x_{47}, x_{48} and x_{49} jointly yield another region of interest.

Table 4.7: First round of the process: KS statistic for $x_1, x_2, x_3, x_{47}, x_{48}, x_{49}$ given the conditions found in each step.

	none	$1.25 < x_1 < 2.75$	$1.81 < x_2 < 2.77$	$1.78 < x_3 < 2.77$
x_1	0.23	-	-	-
x_2	0.27	0.57	-	-
x_3	0.22	0.49	0.56	-
x_{47}	0.33	0.07	0.05	0.05
x_{48}	0.40	0.12	0.03	0.03
x_{49}	0.37	0.08	0.04	0.04

Table 4.8: Fourth round of the process: KS statistic for $x_1, x_2, x_3, x_{47}, x_{48}, x_{49}$ given the conditions found in each step.

	none	$-2.93 < x_{47} < -1.18$	$-2.67 < x_{48} < -1.87$	$-2.39 < x_{49} < -1.62$
x_1	0.23	0.05	0.05	0.04
x_2	0.27	0.09	0.09	0.06
x_3	0.22	0.04	0.04	0.04
x_{47}	0.33	-	-	-
x_{48}	0.40	0.59	-	-
x_{49}	0.37	0.58	0.58	-

4.3 CONCLUSIONS

First of all, although the method is still imperfect, its results are in agreement with the results of a PRIM analysis of an independent sample. This proves that dependent sampling approaches are promising to use for Scenario Discovery and are worth exploring further.

Naturally, this method to derive the combinations of ranges of parameter values yields slightly different results than PRIM. Both the box choice in PRIM and the choice of a threshold value are highly subjective and largely determine the similarity of the values. The threshold value can be increased or decreased to include fewer or more data points, similar to the trade-off between density and coverage in PRIM.

The interpretability of the results is comparable between the dependent and independent approaches. Both yield the combinations of ranges of parameter values that lead to behavior of interest. Intuitive visualizations (like the pairplots with red boxes delineating the regions of interest) have already been developed for the original approach, which can be mimicked by the dependent sampling approach.

An important drawback is the limitation to orthogonal subspaces, similar to PRIM in its early days. Additional research has since been done to resolve this by, for example, performing Principal Component Analysis, rotating the coordinates as a preprocessing step (Dalal et al., 2013).

5 | COMPARING ALGORITHMS

Given the demonstration of the applicability of dependent sampling approaches to Scenario Discovery, this chapter explores a number of MCMC-ABC algorithms which may be applied.

5.1 EXPERIMENTAL SETUP

To compare the performance of these algorithms, each of the components below is varied. The motivation and implementation of each of these are briefly discussed in the remaining of this section.

- Various algorithms, selected from a literature review, contrasted with Latin Hypercube Sampling;
- Four different definitions of the likelihood function, as presented in figure 3.4.
- Various test shapes, adapted from [Lempert et al. \(2008\)](#);

5.1.1 Test shapes

The simple test shapes are taken directly from [Lempert et al. \(2008\)](#). They were originally developed to challenge PRIM and CART in various ways, and will be used here to study the performance of various MCMC algorithms for Scenario Discovery.

The high-dimensional test problems are adapted from the ‘disjoint barrels’ test shape proposed by [Lempert et al. \(2008\)](#) to mimic the envisioned application of the MCMC algorithms, i.e. a large uncertainty space with small, sparse regions of interest. The precise definitions are presented in table 3.2.

5.1.2 Algorithms

In the literature review in section 3.2, three algorithms are selected to test and demonstrate their performance relative to each other and to Latin Hypercube Sampling. The performance of the MCMC algorithms is compared to a Latin Hypercube sample of the uncertainty space, to illustrate the difference between dependent and independent sampling for each type of problem. Adaptive Metropolis is selected to demonstrate the difference between single chain and multi-chain methods. Differential-Evolution Markov Chain contains the core principle of DREAM, but without many of the optimizations implemented in DREAM. The Python implementations of the algorithms are adapted from [Vrugt \(2016\)](#).

To ensure that the results are not caused by an (un)lucky starting position or sample, each experiment is replicated twenty times.

Number of function evaluations

The algorithms are compared given the same number of function evaluations. This implies that the multi-chain methods have a lower number of iterations *per chain*, compared to the single-chain method AM and LHS. The number of function evaluations are used as a measure of computational efficiency.

Latin Hypercube Sampling

A Latin Hypercube Sample is drawn from each test problem. The number of samples is equal to the number of function evaluations in the Monte Carlo methods in order to make a fair comparison. The sampling is performed using the EMA Workbench (Kwakkel, 2017), with the dimensions (x_1, x_2, x_3 , etc) as uncertainties within the bounds identical to the bounds defined for the MCMC methods.

MCMC algorithms

The MCMC algorithms described in section 3.2 are applied to each test problem, combined with the four different definitions of the likelihood function described in section 3.2.5. Each of these experiments is ran for four different numbers of function evaluations, to:

1. Find the most suitable likelihood for each algorithm, the results of which are presented in appendix A. In conclusion, the ‘distance’ likelihood function is used for each algorithm;
2. Compare the performance of the various algorithms, given the performance measures defined in 3.3.

5.2 LATIN HYPERCUBE SAMPLING

5.2.1 Simple test problems

The fractions of cases of interest for the various test shapes and numbers of function evaluations are presented in table 5.1. As expected, the fraction is proportional to the relative size of the region of interest, compared to the full sampled area. The fraction does not change with a higher number of function evaluations.

Table 5.1: Fractions of interest of LH sample of various barrel shapes in 3d

	Function evaluations			
	1000	3000	15000	30000
Upright barrel	0.17	0.18	0.18	0.18
Tilted barrel	0.29	0.28	0.28	0.28
Crossed barrels	0.15	0.16	0.16	0.16
Disjoint barrels	0.21	0.21	0.20	0.20

Characterizing the region of interest using LHS is quite simple. Since the uncertainty space is uniformly sampled, one can use PRIM or CART to find the ranges of uncertain parameters causing cases of interest. For this simple 3d case, it is also possible to directly visualize the shape by colour-coding the cases of interest, as in figure 5.1.

5.2.2 High-dimensional test problems

The fractions of cases of interest in a Latin Hypercube sample of the various test problems are presented in table 5.2. The performance for the problems with the smaller bounds is comparable to performance for the 3d test problems, since the additional dimensions are empty.

Latin Hypercube evidently struggles with the larger bounds: the number of cases of interest drops to a negligibly low number. Increasing the number of function evaluations does not increase the fraction, so an unrealistically high number of function evaluations are required to reliably define the regions of interest.

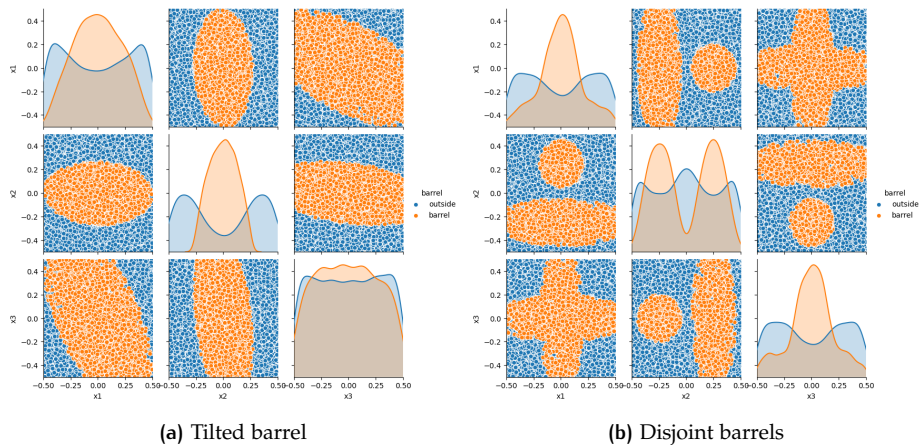


Figure 5.1: Colour-coded pairplots, generated from a Latin Hypercube sample of size 10000.

Table 5.2: Fractions of interest of LH sample of various higher dimensional test problems

	Function evaluations			
	1000	3000	15000	30000
6d, small	0.18	0.20	0.19	0.20
6d, large	<0.01	<0.01	<0.01	<0.01
50d, small	0.11	0.10	0.10	0.10
50d, large	<0.01	<0.01	<0.01	<0.01

5.3 ADAPTIVE METROPOLIS

5.3.1 Simple test problems

Plotting the AM sample confirms that the regions of interest are found, and that the ranges of parameter values are consistent with the Latin Hypercube sample.

The acceptance rate varies slightly between the various test shapes. The tilted barrel yields a higher acceptance rate than the upright barrel, most probably because this shape is larger: there is simply a bigger probability of sampling within the region of interest. Interestingly, the disjoint barrels have a much lower acceptance rate than the upright barrels, while they occupy a similar percentage of the uncertainty space. The MCMC methods have some trouble crossing the space between the regions of interest, since the success of the jump also depends on the jump rate. The fraction of cases of interest is consistently high.

The number of function evaluations does not have an obvious effect on the performance of the algorithm.

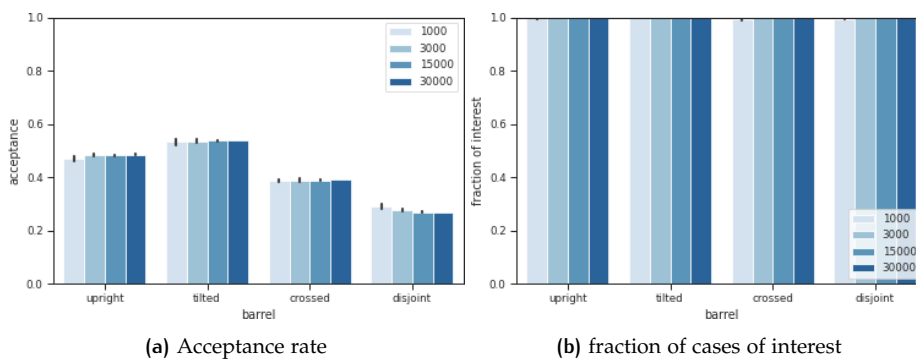


Figure 5.2: Performance of the AM algorithm for the simple test problems

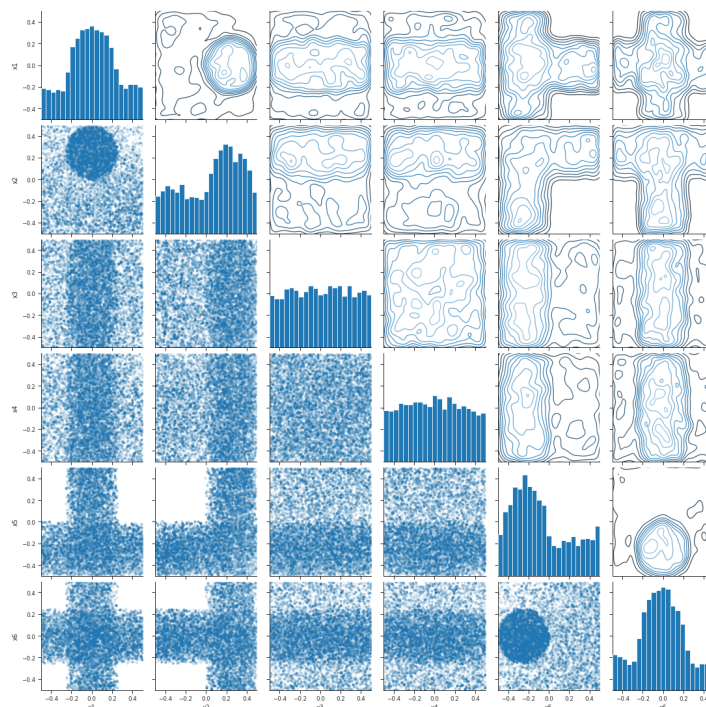


Figure 5.3: The AM algorithm has no trouble finding both input spaces in the small, 6d problem.

5.3.2 High-dimensional test problems

Figure 5.3 demonstrates that, given a suitable starting position and a sufficient number of function evaluations, AM characterizes the region of interest in a small uncertainty space well. The shapes of the regions of interest can easily be distinguished.

Figure 5.4 suggests that in a larger uncertainty space, the AM algorithm may have trouble reaching the region of interest within the given number of function evaluations if the starting position is far away from the region of interest.

Additionally, the AM algorithm fails to find *both* regions of interest, when challenged with a large uncertainty space. When plotting the KS-statistics per dimension for various numbers of function evaluations (while considering the burn-in period and merging the chains), the samples generated by the Adaptive Metropolis algorithm only distinguish one region of interest at a time, since the proposed samples are generated by $p_{i+1} = p_i + \delta$, where δ is generated from a multivariate normal distribution with $mean = 0$ and $covariance = 2.38 \frac{2}{d} \cdot I(d) = 1.04 \cdot I(50)$ (algorithm 3.2), i.e. the jump rate is insufficient to reach the other region of interest once one has been found. Two illustrative AM samples are analysed and displayed in figure 5.5.

5.4 DIFFERENTIAL EVOLUTION MARKOV CHAIN

5.4.1 Simple test problems

Plotting the DE-MC sample confirms that the regions of interest are found, and that the ranges of parameter values are consistent with the Latin Hypercube sample.

Similar to the results of the AM algorithm, the acceptance rate varies slightly between the various test shapes. The fraction of cases of interest is consistently high.

The number of function evaluations does not have an obvious effect on the acceptance rate. The fraction of cases of interest logically increases with increasing number

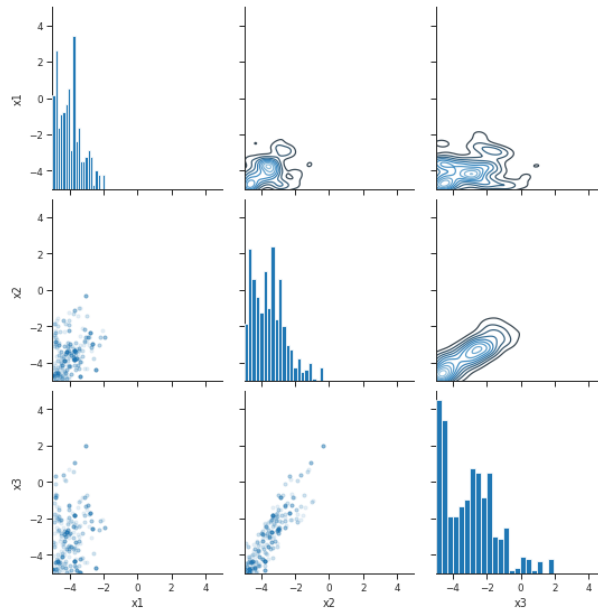
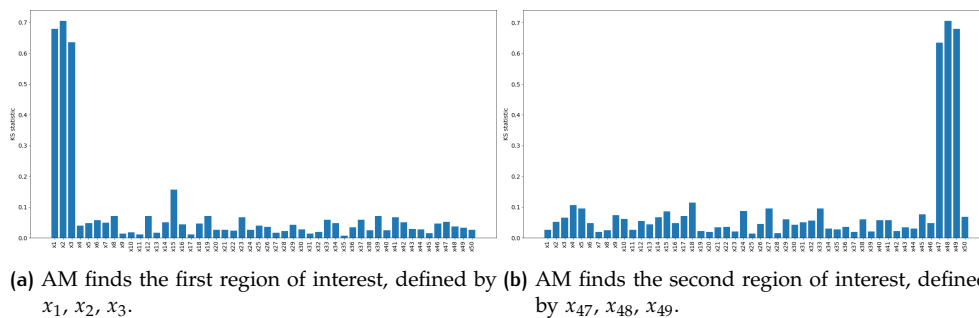


Figure 5.4: The starting position of the AM algorithm is in a corner of the 50-dimensional, large input space, leading to a long trajectory to the region of interest (in the middle of the input space). The number of function evaluations proves insufficient for the algorithm to converge to the region of interest. Only the three dimensions defining the first region of interest are presented here.



(a) AM finds the first region of interest, defined by x_1, x_2, x_3 . **(b)** AM finds the second region of interest, defined by x_{47}, x_{48}, x_{49} .

Figure 5.5: Histograms of the value of the KS statistic for each of the dimensions in the Adaptive Metropolis sample of the 50d, large test problem. The single-chain method only finds one region of interest, because the jumping distance is insufficient to find both.

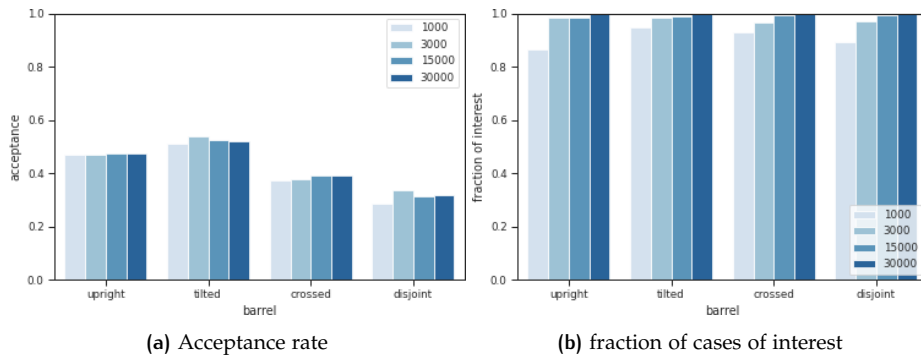


Figure 5.6: Performance of the DE-MC algorithm for the simple test problems

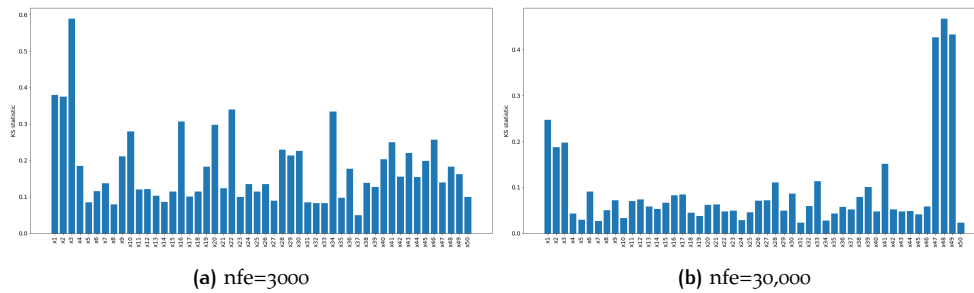


Figure 5.7: Histograms of the value of the KS statistic for each of the dimensions in the Differential Evolution Markov Chain sample of the 50d, large test problem.

of function evaluations, as the algorithm is allowed to converge further to the region(s) of interest. No clear trade-off between efficiency and quality can be distinguished.

5.4.2 High-dimensional test problems

The KS-statistic of the DE-MC samples often exceeds the threshold of 0.05. The dimensions of interest may still be distinguished, but without prior knowledge it would be difficult to rule out some dimensions (e.g. x_{41}). The cumulative probability, displayed in figure 5.8 further illustrates this: compared to the cumulative probability of the DREAM sample (figure 4.4), the blue lines deviate much more from the uniform (red) line. With a lower number of function evaluations (nfe=3000, figure 5.7a), it is impossible to define the regions of interest.

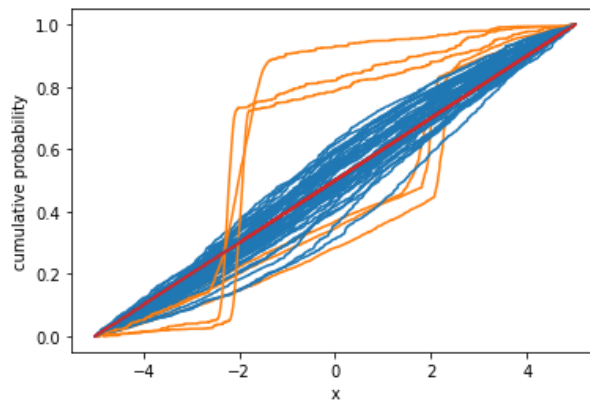


Figure 5.8: Cumulative probability functions of x_1 to x_{50} (DE-MC, nfe=30,000). Dimensions x_1 , x_2 , x_3 , x_{47} , x_{48} and x_{49} are coloured orange for clarity. The cumulative probability of a uniform distribution is displayed in red.

5.5 DIFFERENTIAL EVOLUTION ADAPTIVE METROPOLIS

5.5.1 Simple test problems

Plotting the DREAM sample confirms that the regions of interest are found, and that the ranges of parameter values are consistent with the Latin Hypercube samples.

Similar to the results of the AM and DE-MC algorithms, the acceptance rate varies slightly between the various test shapes. The fraction of cases of interest is consistently high, but not approximately 1, as for AM and DE-MC.

The number of function evaluations does not have an obvious effect on the acceptance rate. The fraction of cases of interest logically increases with an increasing number of function evaluations, as the algorithm is allowed to converge further to the region(s) of interest.

5.5.2 High-dimensional test problems

The DREAM(ABC) sample resembles both regions of interest for every replication, and the regions of interest can be distinguished reasonably well when looking at figure 5.10.

For DREAM(ABC), the number of function evaluations matters for whether or not the KS test yields the desired results. This is illustrated in figure 5.11. AM and DE-MC follow similar behavior.

For a very low number of function evaluations ($nfe=1000$, figure 5.11a), the sample does not represent the full target distribution. Regions of interest cannot be distinguished. For $nfe=3000$ (figure 5.11b), the regions of interest can be distinguished visually, but value of the KS statistic of the irrelevant dimensions is not below the threshold. For $nfe=15,000$ (not pictured) the results are similar to the results presented in figure 4.10 ($nfe=30,000$), so this number of function evaluations is already sufficient.

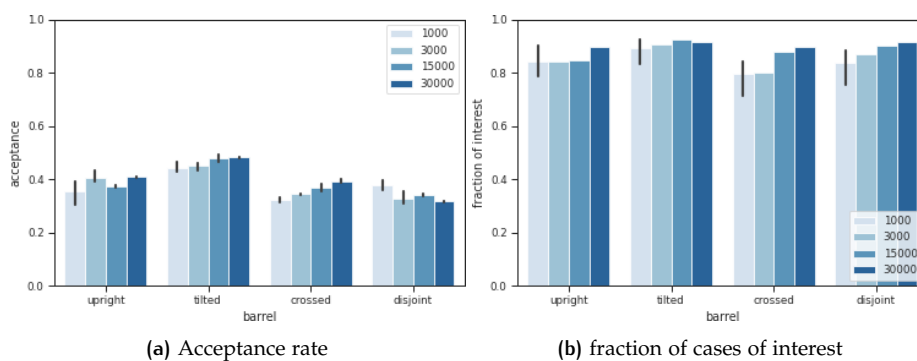


Figure 5.9: Performance of the DREAM algorithm for the simple test problems

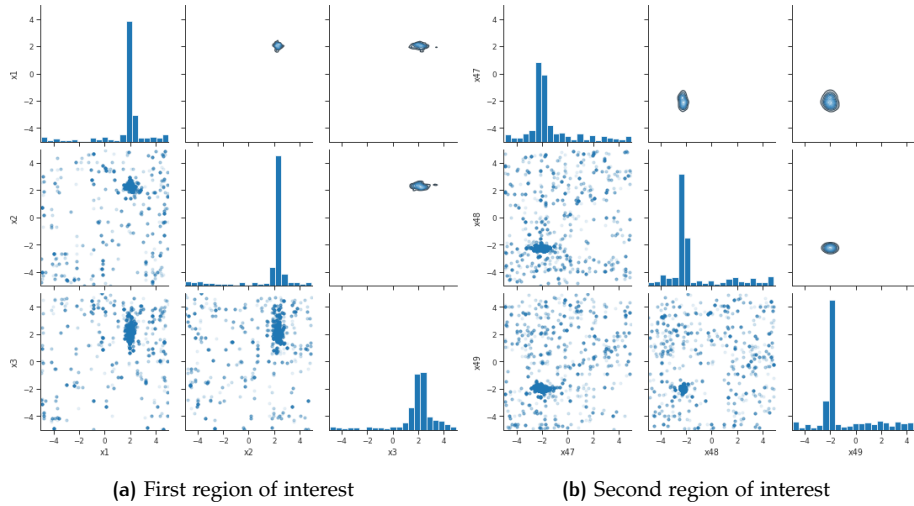


Figure 5.10: DREAM(ABC) sample of 50d test problem, with large bounds

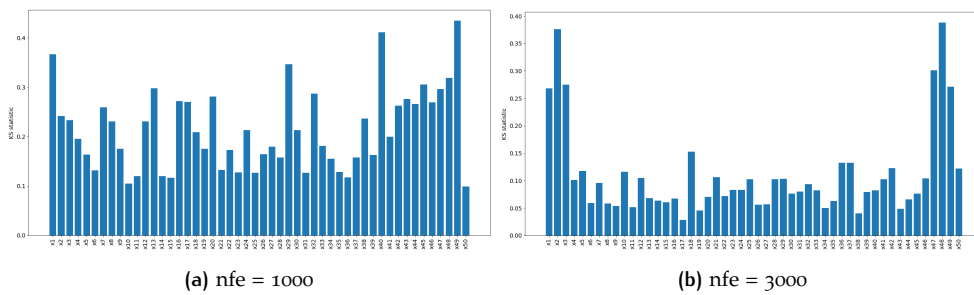


Figure 5.11: Histograms of the value of the KS statistic for each of the dimensions in the DREAM(ABC) sample of the 50d, large test problem. The KS statistic is calculated by comparing to a uniform distribution with bounds $[-5, 5]$.

5.6 COMPARING ALGORITHMS

As evident from figure 5.12, each of the MCMC algorithms outperforms Latin Hypercube with regards to the fraction of cases of interest. The acceptance rates and fractions of cases of interest of the algorithms for the various test problems are presented for a low and high number of function evaluations in figure 5.13. Evidently, the fraction of cases of interest is considerably higher for each of the dependent sampling algorithms than for a Latin Hypercube sample. The difference between the various algorithms is not apparent when applied to the small test problems.

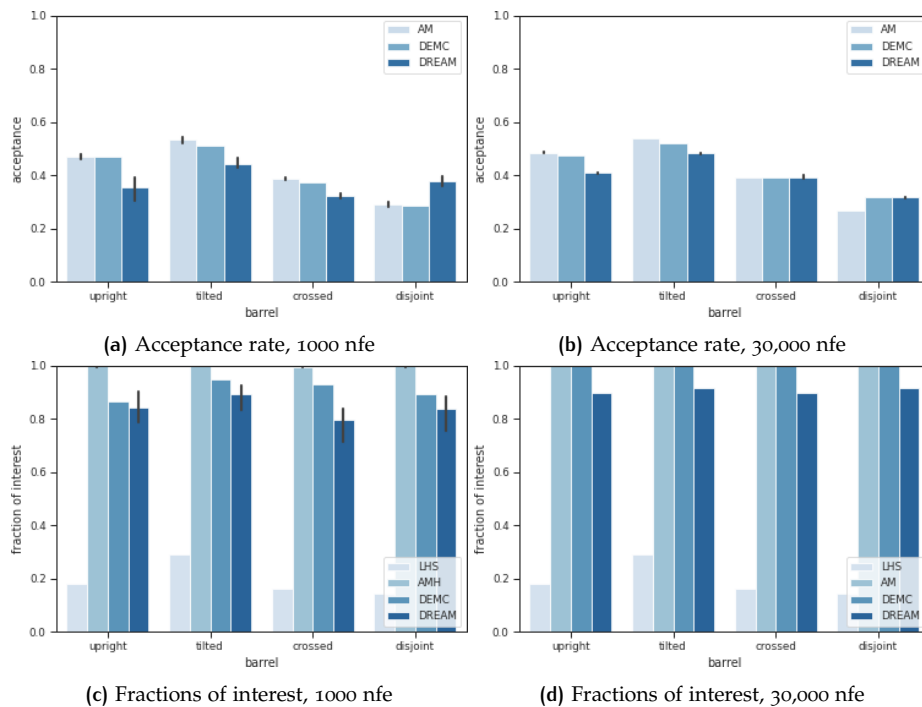


Figure 5.12: Acceptance rates and fractions of interest of the various algorithms for various test shapes in 3d, for 1000 and 30,000 nfe

For the larger test problems, the acceptance rate and fractions of cases of interest do not tell the entire story. Each of the MCMC algorithms occasionally have a *very* low fraction of cases of interest, indicating that the chains do not reach any region of interest within the given nfe. The plots are generated using 20 replications of each experiment, so that this behavior wouldn't skew the results. But this also means that this unreliability is not immediately visible. For DE-MC and DREAM, this happens only at nfe=1,000 and for the '50d - large' test problem. However, for Adaptive Metropolis, it may also happen at nfe=30,000 or for the '6d - small' challenge. This is most likely caused by the way proposal samples are generated in Adaptive Metropolis: by adding a value to the current point ($p_{i+1} = p_i + \delta$). It does not have the jumping features implemented in DE-MC and DREAM.

Besides, an interesting feature of the graphs is that the acceptance rate of the AM algorithm decreases with increasing nfe, while the acceptance rate of DE-MC and DREAM increase with increasing nfe. Simultaneously, the fraction of cases of interest increases for the larger test problems, while it is stable for the test problems with narrower bounds. For the former problems, the algorithms were still progressing towards the posterior target distribution.

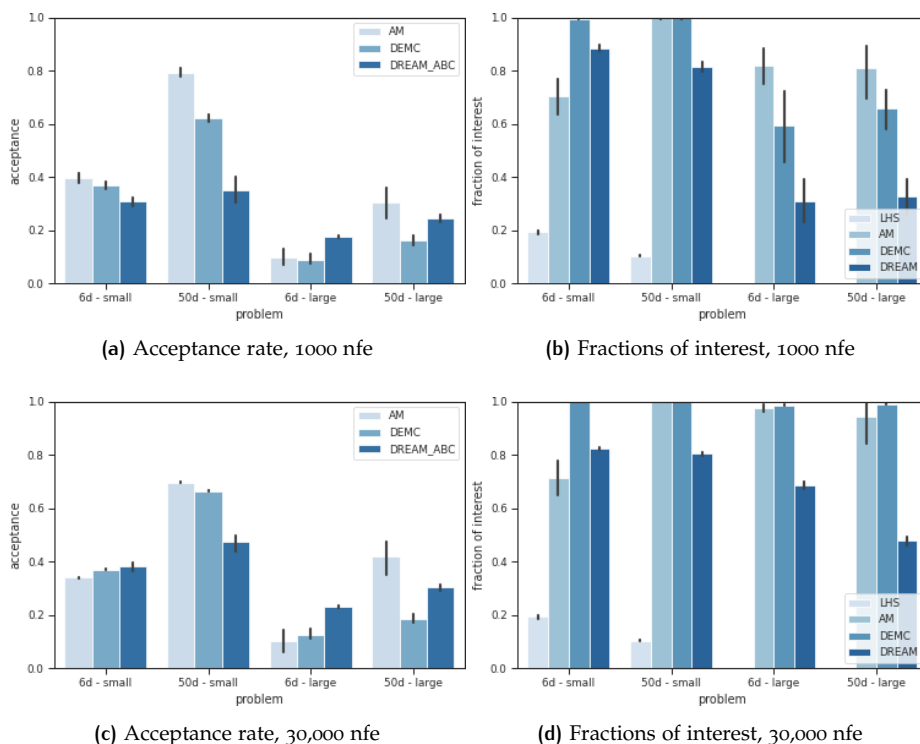


Figure 5.13: Acceptance rates and fractions of interest of the various algorithms for various high-dimensional test shapes, for 1000 and 30,000 nfe

5.7 CONCLUSIONS

From the experiments described in the previous sections, a number of conclusions can be drawn, which answer the fourth subquestion, as discussed in the introduction.

Each of the MCMC algorithms outperforms Latin Hypercube, especially for high-dimensional problems with sparse regions of interest. This is most evident for the test problems with expanded bounds, where a Latin Hypercube sample is insufficient to characterize the region of interest using PRIM given a reasonable number of function evaluations.

With regards to the relative performance of the various algorithms, the number of function evaluations seems to impact each MCMC-ABC algorithm differently: with a higher nfe, the fraction of cases of interest increases for DE-MC, stays approximately 100% for AM and is stable around 90% for DREAM. The choice of likelihood function is determined based on experiments on the small and large test problems. For each algorithm, the function that decreases linearly with increasing distance from the boundary of the region of interest leads to the highest fraction of cases of interest. For the remainder of the experiments and conclusions, this setting is used for each algorithm.

The Adaptive Metropolis algorithm is not very reliable. Often, it is not able to adequately represent the regions of interest, as demonstrated in figure 5.4. Additionally, if the regions of interest are further apart, AM gets stuck in a single region of interest and fails to explore the uncertainty space.

DE-MC and DREAM consistently perform well, yielding a high fraction of cases of interest and a reasonable acceptance rate. DE-MC and DREAM consistently find all regions of interest. DREAM has a higher acceptance rate than DE-MC for the larger test problems, most probably due to DREAMs subspace sampling, since only a few dimensions are actually of interest.

6

CONCLUSIONS & DISCUSSION

6.1 ANSWERING THE RESEARCH QUESTION

This final chapter addresses the research question posed in chapter 1: ‘How can dependent sampling methods be used for effective Scenario Discovery to characterize high-dimensional problems with sparse regions of interest?’ To do so, the sub-questions are briefly discussed, after which the proposed methodology for Scenario Discovery is presented, answering the research question.

1. Which dependent sampling methods may be employed for Scenario Discovery?

Since a formal definition of the likelihood function is not possible for complex simulation models, sampling should be performed using Approximate Bayesian Computation (ABC), where proposal samples are always accepted when their performance exceeds the performance of the current point and always rejected when their performance is worse. Normal Markov Chain Monte Carlo (MCMC) methods, which reject samples based on a probability proportional to the relative performance, are not applicable because they require a formal likelihood function.

The specific algorithms have been selected based on the availability of a Python implementation, their respective contribution to the understanding of MCMC methods for Scenario Discovery, and to what extent they support the academic argument. Besides Latin Hypercube Sampling, Adaptive Metropolis, Differential Evolution Markov chain and Differential Evolution Adaptive Metropolis are applied.

2. Which measures of performance can be used to compare various algorithms for Scenario Discovery?

The selected performance measures describe both the efficiency with which the sample is collected and the quality of the sample. The efficiency is quantified by collecting the acceptance rate of the MCMC implementations of ABC, i.e. the fraction of accepted proposals. The acceptance rate gives insight in how many of the function evaluations are productive. Secondly, experiments are conducted with various numbers of function evaluations, to qualitatively assess whether the trade-off between quality and speed of analysis varies between algorithms.

Furthermore, the quality of the sample is determined by qualitative measures. The sampling approach should find *all* of the regions of interest. The known test problems used in this thesis allow for easy confirmation, but this is not possible when applying MCMC-ABC for Scenario Discovery on unknown problems.

Besides, the fraction of cases of interest is calculated for each of the sampling approaches (independent and dependent). This measure indicates how well the sample may be analysed to provide insight in the region(s) of interest. In section 5.2.2, it is demonstrated that this fraction is problematically low for an independent sample of large uncertainty spaces with sparse regions of interest. A sample with a low fraction of cases of interest cannot be adequately analysed and is therefore not suitable for Scenario Discovery. This also partly reflects the quality of the sample.

Finally, the intuitive interpretability of the results is paramount to use the sampling approach for Scenario Discovery in a decision-making context. Interpretability is a rather vague performance measure. [Lempert et al. \(2008\)](#) used the number of boxes

and the number of restricted dimensions (in a way, the number of ‘things’ communicated to a decision-maker) as a proxy. These performance measures are not appropriate for the results of a dependent sampler. Instead, chapter 4 develops a methodology to derive the same information from an MCMC sample as PRIM would, proving that the interpretability is at least similar for both methods.

3. *How can combinations of input parameters that are highly predictive of the behavior of interest be derived from an MCMC sample?*

Independent sampling approaches yield intuitive plots and easily interpreted ranges of parameter values. Its efficacy in a decision-making context has been repeatedly proven.

The interpretation of the results of dependent sampling in the context of Scenario Discovery is not straight-forward. These MCMC algorithms yield a posterior target distribution, rather than neat ranges of parameter values causing the behavior of interest. Chapter 4 demonstrates the extraction of the combinations of parameter values causing certain behavior of interest.

While the methodology is still a work in progress, it proves the applicability of the approach for Scenario Discovery and provides additional methods to assess the performance of the various algorithms.

4. *What is the relative efficacy of dependent sampling algorithms for high-dimensional problems with sparse regions of interest, compared to an independent sampling approach?*

Chapter 5 compares algorithms by experimenting on increasingly challenging test problems. Even for small test problems, where the region of interest is relatively large, each of the MCMC algorithms outperforms Latin Hypercube with regards to the fraction of cases of interest. The gap is even larger for high-dimensional problems with sparse regions of interest. A Latin Hypercube sampler cannot adequately characterize the region of interest using PRIM given a reasonable number of function evaluations.

The three MCMC-ABC algorithms applied in the experiments yield varying results. The Adaptive Metropolis algorithm is not very reliable. Often, it is not able to adequately represent the regions of interest, as demonstrated in figure 5.5. DE-MC consistently achieves a fraction of cases of interest close to 1, and a reasonably high acceptance rate. However, the resulting sample contains more clutter and is therefore less applicable to Scenario Discovery. Due to the features in DREAM that force exploration of the input space, the fraction of cases of interest of the DREAM samples are consistently lower than for DE-MC. However, the resulting sample represents the true posterior distribution and is readily analysed to derive the combinations of parameter values that cause a behavior of interest.

6.1.1 Proposed methodology

To answer the main research question, this section describes the proposed methodology for Scenario Discovery using a dependent sampling approach.

Problems that fulfill the following three criteria cannot be adequately analysed using an independent sampling approach and should therefore be considered to be characterized using this methodology:

- Large uncertainty space, both with regards to the number of dimensions and, most importantly, the bounds;
- Sparse regions of interest: a Latin Hypercube sample would yield a problematically low number of cases of interest;
- A long run time of the model, i.e. it is unfeasible to compensate the former points by drawing a larger sample.

Based on the answers to the subquestions, I recommend the following approach for Scenario Discovery using depending sampling:

1. Determine the ranges of parameter values to be sampled
2. Define a likelihood function that transforms the model outputs to a single value which is highest when it is of interest and decreases when it is of decreasing interest. This is based on a threshold, similar to the ‘labeling’ step in the original Scenario Discovery approach.
3. Choose an MCMC-ABC sampler. I used an adapted version of pyDREAM (Shockley et al., 2017) with good results, but other algorithms exist, including, but not limited to STAN (Carpenter et al., 2017) and pyMC3 (Salvatier et al., 2016).
4. Draw a few thousand samples using the MCMC-ABC sampler of choice and evaluate the convergence, or directly evaluate the results by collecting some initial statistics and visual inspection. If the algorithm has not converged, continue sampling until it has.
5. Calculate the Kolmogorov-Smirnov statistic for each dimension to determine the parameters driving the behavior of interest.
6. Derive the combinations of ranges of parameter values driving the behavior of interest by deriving the ranges of inputs for each dimension and iteratively selecting subsets of the data, following the steps proposed in section 4.2.2.

6.2 DISCUSSION

6.2.1 Predetermined definitions of behavior of interest

One important limitation to the approach is the need to define behavior of interest *before* sampling. When using a Latin Hypercube sample for Scenario Discovery, this sample can be reused for other purposes, i.e. one can experiment with different definitions of failure/success, or run additional analyses. Instead, in MCMC, the behavior of interest determines where the sampling is concentrated. The sample resembles the posterior of that behavior of interest and can therefore not be reused for different purposes.

6.2.2 Computational considerations

Apart from the comparison of the efficacy of the various sampling algorithms as a measure of performance through proxy statistics, the computational considerations for choosing one paradigm over the other is not thoroughly discussed in the body of this thesis. However, when discussing the need for new sampling approaches for large, complex models, it is illogical to ignore these.

Resuming sampling

Firstly, an advantage of MCMC methods over Latin Hypercube is the possibility of continuing the sampling process when it has been interrupted. One can test whether the number of samples is sufficient, and if not, simply feed the chains as history to the algorithm again. Theoretically, it is possible to double or quadruple the number of points in a Latin Hypercube Sample, by drawing n points per subspace instead of one, but this is rather inflexible.

Parallelization

Secondly, MCMC methods are more difficult to run in parallel. Independent sampling methods could fully parallelized, significantly speeding up experimentation if resources are endless. In MCMC, the various chains may be ran in parallel. If the model is sufficiently complex, so that distributed or parallel simulation of the model is beneficial, this is also possible. However, as candidate points are generated from the current position, it is not possible to further parallelize the experimentation.

6.2.3 Comparing computational efficiency

Lastly, the convergence of the MCMC algorithms is not readily determined. The most frequently used convergence diagnostic for multi-chain methods is the Gelman-Rubin \hat{R} (r-hat) statistic (Carpenter et al., 2017; Vrugt, 2016; Shockley et al., 2017; Salvatier et al., 2016), which is calculated from the variance within (W) and between (B) the m number of chains of length n , for each parameter of interest θ (Gelman and Rubin, 1992):

$$B = \frac{n}{m-1} \sum_{j=1}^m (\bar{\theta}_{.j} - \bar{\theta}_{..})^2$$

$$W = \frac{1}{m} \sum_{j=1}^m \left[\frac{1}{n-1} \sum_{i=1}^n (\theta_{ij} - \bar{\theta}_{.j})^2 \right]$$
(6.1)

The posterior variance of θ can be estimated by:

$$\hat{\text{Var}}(\theta|y) = \frac{n-1}{n}W + \frac{1}{n}B$$
(6.2)

The Gelman-Rubin diagnostic can then be monitored by tracking:

$$\hat{R} = \sqrt{\frac{\hat{\text{Var}}(\theta|y)}{W}}$$
(6.3)

For MCMC methods (and ABC methods specifically), the algorithm is believed to have converged when \hat{R} is below 1.2 for each chain (Vrugt et al., 2009; Sadegh and Vrugt, 2014).

\hat{R} is used in pydream, and is suggested to use in combination with pydream(ABC) (Vrugt, 2016). Additionally, both STAN and pyMC3 use \hat{R} , the Gelman Rubin convergence diagnostic. Theoretically, it should be possible to use this diagnostic to assess the convergence. In comparison, one can also only assess whether a Latin Hypercube sample is sufficient after completing the sampling process. This is a limitation of both approaches, but MCMC algorithms are more easily continued, rather than repeated with a higher number of iterations.

Because the convergence of the algorithms is not assessed in the experiments in this thesis, the computational efficiency cannot be directly compared. Instead, some experimentation is done with various numbers of function evaluations to gain some insights.

6.2.4 Discrete and categorical parameters

Furthermore, similar to the original Scenario Discovery approach, MCMC methods are able to handle discrete and categorical parameters values. The sampled real value is floored to the nearest integer, which may also be interpreted as a category, as demonstrated in Vrugt and Ter Braak (2011).

6.2.5 Finding all regions of interest

With regards to the performance measures, it is difficult to assess whether *all* regions of interest have been discovered. For the known problems this can be checked, and it doesn't seem to be an issue. However, it is important to stay aware of, and potentially develop some diagnostic for. Independent sampling methods are less vulnerable to this, since the entire uncertainty space is always sampled.

6.3 IMPACT OF RESEARCH

This thesis presents an approach to Scenario Discovery for complex problems with a large input space with sparse regions of interest using an MCMC-ABC sampler. This development allows for the analysis of a wide array of models which previously required preprocessing to select or extract the relevant uncertain input parameters.

6.4 FUTURE DIRECTIONS OF RESEARCH

To finalize the approach to Scenario Discovery using dependent sampling and incorporate this in the standard workflow, more time should be dedicated to transforming the results into intuitive visualizations and advice helpful to decision-makers.

To further demonstrate the added value, it should be applied to a suitable case study. The test problems presented in this thesis share characteristics found in complex problems, but lack complex dynamics and interactions.

Additionally, application to a real model may develop understanding of the limits of the approach. Since convergence is difficult to assess, it is important to have an idea of the limitations, to avoid wasting time sampling to no avail. For example, the MCMC-ABC sampler may struggle with problems with many distinct regions of interest, since it has to jump. Or one may find a limit to the number of dimensions that can be effectively sampled.

Furthermore, ways to find non-orthogonal subspaces with PRIM have been explored [Dalal et al. \(2013\)](#). The method as proposed in this thesis exclusively yields orthogonal subspaces. Similar efforts should be made for this method to rival the versatility of PRIM.

Besides, different likelihood-free dependent sampling approaches should be explored. DREAM(ABC) yields promising results, but for real-world applications it is worth it to test other approaches, which potentially perform better with regards to computational efficiency or quality of the sample.

BIBLIOGRAPHY

- Agusdinata, D. B. (2008). *Exploratory Modeling and Analysis: A Promising Method to Deal with Deep Uncertainty*. PhD thesis, Delft University of Technology.
- Bradfield, R., Wright, G., Burt, G., Cairns, G., Van Der Heijden, K., and Heijden, K. V. D. (2005). The origins and evolution of scenario techniques in long range business planning. *Futures*, 37(8):795–812.
- Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis.
- Bryant, B. P. and Lempert, R. J. (2010). Thinking inside the box: A participatory, computer-assisted approach to scenario discovery. *Technological Forecasting and Social Change*, 77(1):34–49.
- Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1).
- Chu, W., Yang, T., and Gao, X. (2014). Comment on “High-dimensional posterior exploration of hydrologic models using multiple-try DREAM (ZS) and high-performance computing” by Eric Laloy and Jasper A. Vrugt. *Water Resources Research*, 50(3):2775–2780.
- Dalal, S., Han, B., Lempert, R., Jaycocks, A., and Hackbarth, A. (2013). Improving scenario discovery using orthogonal rotations. *Environmental Modelling & Software*, 48:49–64.
- Diggle, P. J. and Gratton, R. J. (1984). Monte Carlo Methods of Inference for Implicit Statistical Models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(2):193–212.
- Fisher, R. A. (1935). *The design of experiments*. Oliver & Boyd, Oxford, England.
- Friedman, J. and Fisher, N. (1999). Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143.
- Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):467–511.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press.
- Groves, D. G. and Lempert, R. J. (2007). A new analytic method for finding policy-relevant scenarios. *Global Environmental Change*, 17(1):73–85.
- Haario, H., Saksman, E., and Tamminen, J. (2001). An Adaptive Metropolis Algorithm. *Bernoulli*, 7(2):223.
- Haasnoot, M., Middelkoop, H., Beek, E. V., and Deursen, W. P. A. V. (2011). Management Strategies for an Uncertain Future. *Sustainable Development*, 19:369–381.
- Halim, R. A., Kwakkel, J. H., and Tavasszy, L. A. (2016). A scenario discovery study of the impact of uncertainties in the global container transport system on European ports. *Futures: the journal of policy, planning and futures studies*, 81:148–160.

- Kim, H., Robert, C. P., and Casella, G. (2000). *Monte Carlo Statistical Methods*, volume 42.
- Kwadijk, J. C. J., Haasnoot, M., Mulder, J. P. M., Hoogvliet, M. M. C., Jeuken, A. B. M., van der Krogt, R. A. A., van Oostrom, N. G. C., Schelfhout, H. A., van Velzen, E. H., van Waveren, H., and de Wit, M. J. M. (2010). Using adaptation tipping points to prepare for climate change and sea level rise: a case study in the Netherlands. *WIREs Climate Change*, 1(5):729–740.
- Kwakkel, J. H. (2017). The Exploratory Modeling Workbench: An open source toolkit for exploratory modeling, scenario discovery, and (multi-objective) robust decision making. *Environmental Modelling and Software*, 96:239–250.
- Kwakkel, J. H. (2019). A generalized many - objective optimization approach for scenario discovery. *Futures and Foresight Science.*, 1(2):1–13.
- Kwakkel, J. H., Auping, W. L., and Pruyt, E. (2013). Dynamic scenario discovery under deep uncertainty: The future of copper. *Technological Forecasting and Social Change*, 80(4):789–800.
- Kwakkel, J. H. and Cunningham, S. C. (2016). Improving scenario discovery by bagging random boxes. *Technological Forecasting and Social Change*, 111:124–134.
- Laloy, E. and Vrugt, J. A. (2012). High-dimensional posterior exploration of hydrologic models using multiple-try DREAM (ZS) and high-performance computing. *Water Resources Research*, 48(1):2775–2780.
- Lempert, R. J., Bryant, B. P., and Bankes, S. C. (2008). Comparing Algorithms for Scenario Discovery. Technical Report January, RAND Corporation.
- Lempert, R. J., Groves, D. G., Popper, S. W., and Bankes, S. C. (2006). A general, analytic method for generating robust strategies and narrative scenarios. *Management Science*, 52(4):514–528.
- Lempert, R. J., Popper, S. W., and Bankes, S. C. (2003). *Shaping the Next One Hundred Years: New Methods for Quantitative, Long-Term Policy Analysis*.
- Massey, F. J. (1951). The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68–78.
- Mathworks, I. (2020). One-sample Kolmogorov-Smirnov test.
- McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21(2):239.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*.
- Moksnes, N., Rozenberg, J., Broad, O., Taliotis, C., Howells, M., and Rogner, H. (2019). Determinants of energy futures — a scenario discovery method applied to cost and carbon emission futures for South American electricity infrastructure. *Environmental Research Communications*, 1:1–25.
- Parzen, E. (1962). On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33(3):1065–1076.
- Price, K., Storn, R., and Lampinen, J. (2005). *Differential Evolution-A Practical Approach to Global Optimization*, volume 141.

- Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function. *Ann. Math. Statist.*, 27(3):832–837.
- Sadegh, M. and Vrugt, J. A. (2014). Approximate Bayesian Computation using Markov Chain Monte Carlo simulation: DREAM(ABC). *Water Resources Research*, 50(8):6767–6787.
- Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. (2016). Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*.
- Shockley, E. M., Vrugt, J. A., and Lopez, C. F. (2017). PyDREAM: high-dimensional parameter inference for biological models in python. *Bioinformatics*, 34(4):695–697.
- Steinmann, P., Auping, W. L., and Kwakkel, J. H. (2020). Behavior-based scenario discovery using time series clustering. *Technological Forecasting and Social Change*, 156:120052.
- Storn, R. and Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359.
- Sunnåker, M., Busetto, A. G., Numminen, E., Corander, J., Foll, M., and Dessimoz, C. (2013). Approximate Bayesian Computation. *PLOS Computational Biology*, 9(1):1–10.
- Ter Braak, C. J. (2006). A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: Easy Bayesian computing for real parameter spaces. *Statistics and Computing*, 16(3):239–249.
- ter Braak, C. J. F. and Vrugt, J. A. (2008). Differential Evolution Markov Chain with snooker updater and fewer chains. *Statistics and Computing*, 18(4):435–446.
- Turner, B. M. and Van Zandt, T. (2012). A tutorial on approximate Bayesian computation. *Journal of Mathematical Psychology*, 56(2):69–85.
- Vrugt, J. A. (2016). Markov chain Monte Carlo simulation using the DREAM software package: Theory, concepts, and MATLAB implementation. *Environmental Modelling and Software*, 75:273–316.
- Vrugt, J. A. and Beven, K. J. (2018). Embracing equifinality with efficiency: Limits of Acceptability sampling using the DREAM(LOA) algorithm. *Journal of Hydrology*, 559:954–971.
- Vrugt, J. A. and Sadegh, M. (2013). Toward diagnostic model calibration and evaluation: Approximate bayesian computation. *Water Resources Research*, 49(7):4335–4345.
- Vrugt, J. A. and Ter Braak, C. J. (2011). DREAM(D): An adaptive Markov Chain Monte Carlo simulation algorithm to solve discrete, noncontinuous, and combinatorial posterior parameter estimation problems. *Hydrology and Earth System Sciences*, 15(12):3701–3713.
- Vrugt, J. A., Ter Braak, C. J., Diks, C. G., Robinson, B. A., Hyman, J. M., and Higdon, D. (2009). Accelerating Markov chain Monte Carlo simulation by differential evolution with self-adaptive randomized subspace sampling. *International Journal of Nonlinear Sciences and Numerical Simulation*, 10(3):273–290.
- Walker, W. E., Lempert, R. J., and Kwakkel, J. H. (2013). *Deep Uncertainty*, pages 395–402. Springer US, Boston, MA.
- Weber, J. (2019). Design of Experiments.



THE SHAPE OF THE LIKELIHOOD FUNCTION

A.1 ADAPTIVE METROPOLIS

A.1.1 Simple test problems

To draw conclusions regarding its performance, twenty duplicates of the experiments are performed to visualize the spread of performance.

The results of these replications for the simple test problems are visualized in figure A.1. The Adaptive Metropolis algorithm shows a large variance in the acceptance rate for each of the likelihood functions. However, this does not affect the corresponding fraction of interest, which is consistently very high (ranging from 0.975 to 1). In these graphs, the entire chain is used. The performance increases even further when using a burn-in period of 10% of the sample. Then, the fraction of interest is 1 for each case.

A.1.2 High-dimensional test problems

Since the comparison of likelihood functions for the AM algorithm for the 3d barrels was inconclusive, a similar experiment was conducted using the 50d implementation with a large uncertainty space. The results of the twenty replications performed of this experiment are presented in figure A.2.

The acceptance rates are comparable for 15,000-30,000 nfe are comparable across likelihood functions. However, for the hyperbolic tangent shapes, the acceptance rates are often (close to) 100%. These extraordinarily large values for the acceptance rates are accompanied by fractions of cases of interest of (close to) zero. This can be explained by the shape of the likelihood function: further away from the region of interest, the likelihood converges to zero. Therefore, the chain does not ‘know’ in which direction to move to find the regions of interest. In contrast, using the distance function, the chain always moves towards the region of interest (or the proposed sample is not accepted).

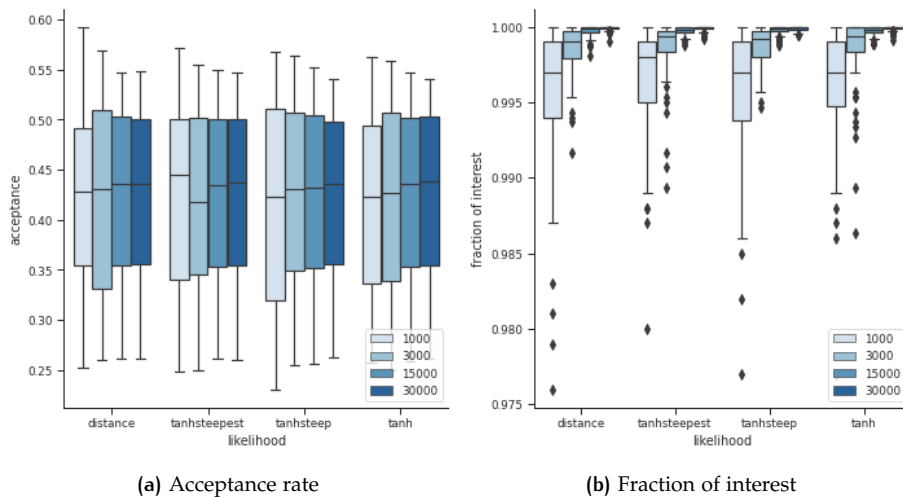


Figure A.1: Performance of the Adaptive Metropolis algorithm for various likelihood functions

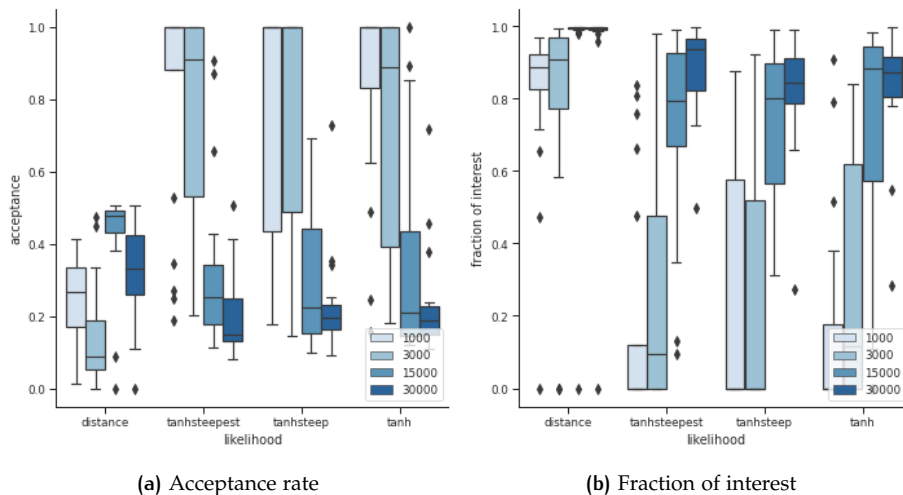


Figure A.2: Performance of the AM algorithm for various likelihood functions for the 50d, large box test problem

The ‘distance’ likelihood function will therefore be used in combination with the AM algorithm for the remainder of the experiments.

A.2 DIFFERENTIAL EVOLUTION MARKOV CHAIN

A.2.1 Simple test problems

For the simple test problems, the performance DE-MC algorithm is not very dependent on the likelihood function. The fraction of interest is generally high: it varies between 86% and 100%, primarily depending on the number of function evaluations, but also on the run and the specific barrel shape. Similar experiments will be ran for the high-dimensional test shapes, to examine whether the shape of the likelihood function has an effect when dealing with more sparse regions of interest.

A.2.2 High-dimensional test problems

Figure A.4 displays the acceptance rates of the DE-MC algorithm for the 50d test problem with expanded bounds. The acceptance rate is similar for each likelihood function. The variance decreases with increasing nfe, but the mean does not change significantly. The fraction of interest, however, converges close to 1 with increasing nfe for each likelihood function shape. The ‘distance’ function shows the highest fraction of interest and the highest reliability for each nfe. For the smaller problem, the fraction of interest is 1 or close to 1 for any of the likelihood functions.

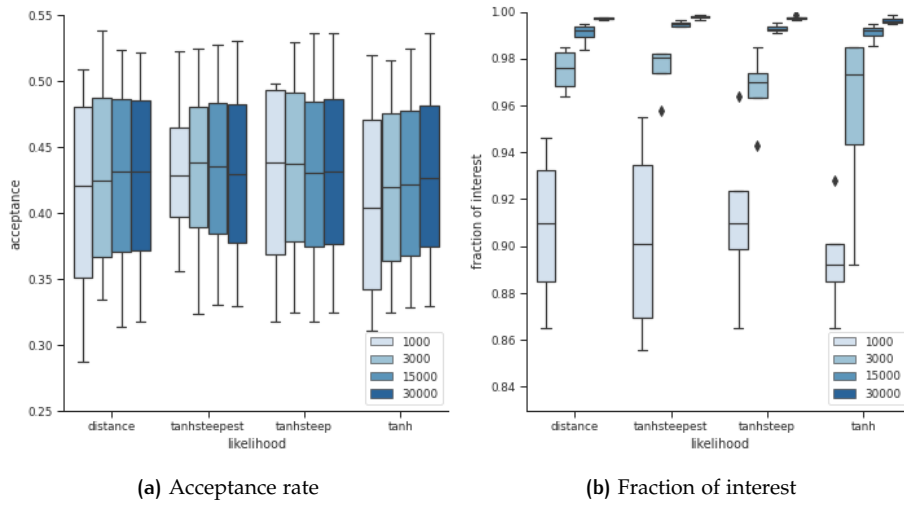


Figure A.3: Performance of the DE-MC algorithm for various likelihood functions

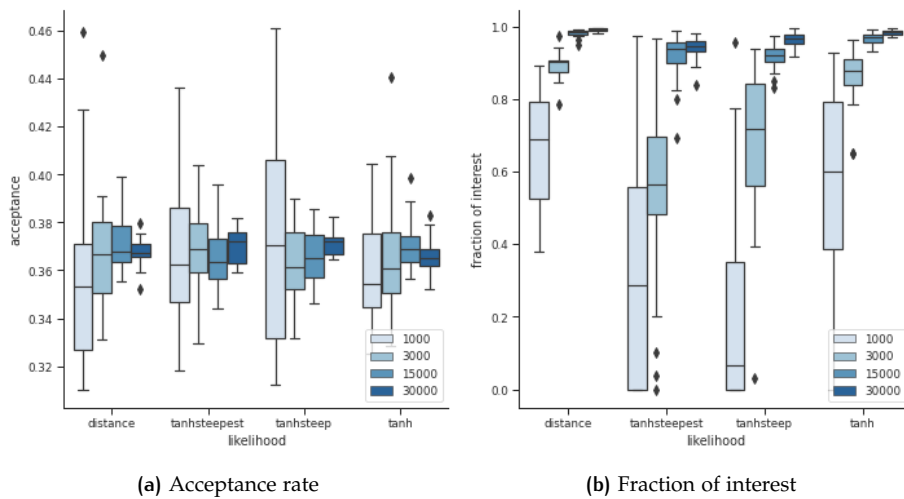


Figure A.4: Performance of the DE-MC algorithm for various likelihood functions for the 50d, large box test problem

A.3 DIFFERENTIAL EVOLUTION ADAPTIVE METROPOLIS

The acceptance rate is not significantly affected by the shape of the likelihood function. However, the fraction of cases of interest is highest for the ‘distance’ function, and increases with increasing nfe. This is most likely due to the way DREAM generates the next sample, i.e. from the difference between other chains. Therefore, the sample can jump through the uncertainty space, while the Adaptive Metropolis algorithm generates its next sample by adding some randomly generated ‘delta’ value to the current sample.

Therefore, all subsequent experiments with the DREAM(ABC) algorithm will be performed using the ‘distance’ likelihood function. This is confirmed by experiments with the largest test problems (not displayed here).

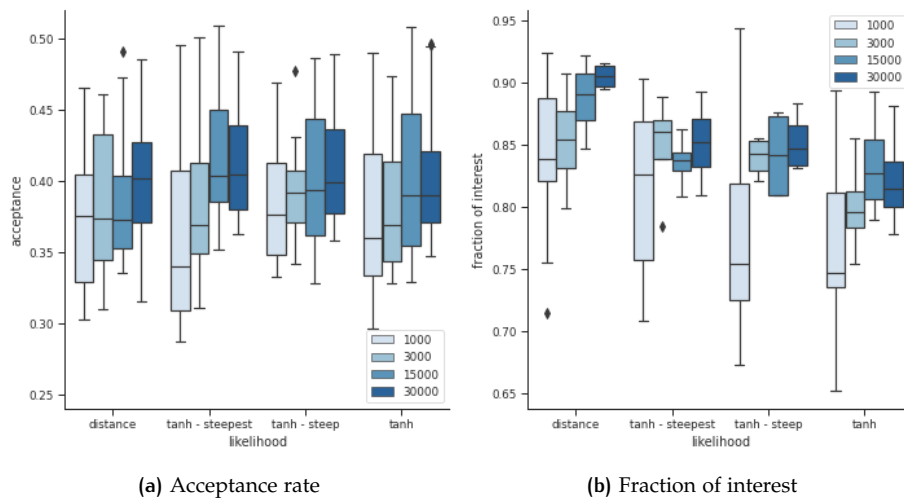


Figure A.5: Performance of the DREAM(ABC) algorithm for various likelihood functions

A.4 CHOOSING THE LIKELIHOOD FUNCTION

Using the performance measures acceptance rate and the fraction of cases of interest, the most suitable likelihood function was selected for each algorithm. The ‘distance’ likelihood function leads to the best performance for each algorithm for each test shape and will therefore be used for the experiments conducted in this thesis.

