Springer Proceedings in Mathematics & Statistics

Alexander Keller *Editor*

Monte Carlo and Quasi-Monte Carlo Methods

MCQMC 2020, Oxford, United Kingdom, August 10–14

Springer

Springer Proceedings in Mathematics & Statistics

Volume 387

This book series features volumes composed of selected contributions from workshops and conferences in all areas of current research in mathematics and statistics, including data science, operations research and optimization. In addition to an overall evaluation of the interest, scientific quality, and timeliness of each proposal at the hands of the publisher, individual contributions are all refereed to the high quality standards of leading journals in the field. Thus, this series provides the research community with well-edited, authoritative reports on developments in the most exciting areas of mathematical and statistical research today.

More information about this series at https://link.springer.com/bookseries/10533

Alexander Keller
Editor

Monte Carlo and Quasi-Monte Carlo Methods

MCQMC 2020, Oxford, United Kingdom, August 10–14

Springer

Editor
Alexander Keller
NVIDIA
Berlin, Germany

ISSN 2194-1009 ISSN 2194-1017 (electronic)
Springer Proceedings in Mathematics & Statistics
ISBN 978-3-030-98318-5 ISBN 978-3-030-98319-2 (eBook)
https://doi.org/10.1007/978-3-030-98319-2

Mathematics Subject Classification: 11K45, 11K38, 11K36, 65D30, 65C05, 65C10, 11K16, 91G60, 65-04, 62G07, 68Q17, 65R20, 65C40, 62D05, 68T07

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume represents the refereed proceedings of the 14th International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing which was held online, August 10–14, 2020. The program of the conference consisted of 97 recorded regular talks featured on the MCQMC presentations YouTube channel. Highlights include the invited plenary talks presented live by Yves Atchadé (Boston University), Jing Dong (Columbia University), Pierre L'Ecuyer (University of Montreal), Mark Jerrum (Queen Mary University London), Peter Kritzer (RICAM Linz), Thomas Müller (NVIDIA), David Pfau (Google DeepMind), Claudia Schillings (University of Mannheim), Mario Ullrich (JKU Linz), and the tutorials by Aretha Teckentrup (Edinburgh) and Fred Hickernell (IIT). While the MCQMC conference regularly attracts between 180 and 240 attendees, more than 600 participants registered for the online version of MCQMC 2020.

The articles in this volume were carefully screened and cover both the theory and the applications of Monte Carlo and quasi-Monte Carlo methods in scientific computing. We thank the anonymous reviewers for their reports and many others who contributed enormously to the excellent quality of the conference presentations and to the high standards for publication in these proceedings by reviewing the abstracts and manuscripts that were submitted.

The next International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing (MCQMC 2022) will be hosted by the Johannes Kepler University (JKU) and the Johann Radon Institute for Computational and Applied Mathematics (RICAM) in Linz, Austria, in July 2022.

Berlin, Germany
November 2021

Alexander Keller

Acknowledgements

The University of Oxford organizing committee had been looking forward to welcoming you all to Oxford. Unfortunately of course, the coronavirus pandemic has completely ruined all plans. Meeting in person has not been possible; indeed the United Kingdom has been one of the worst affected countries with all of the University of Oxford organizing committee significantly impacted in their jobs.

We appreciate the assistance and sponsoring of the International Centre for Mathematical Sciences (ICMS) in hosting the plenary talks and tutorials. We thank all contributors, who recorded their presentations, and the Berlin organizing committee, who took care of the MCQMC presentations YouTube channel and the conference program booklet. We are very grateful for the support from our sponsors the Mathematical Institute, University of Oxford, the Department of Statistics, University of Oxford, and the Engineering and Physical Sciences Research Council (EPSRC). Last but not least, we thank the scientific committee for arranging an exceptional conference program.

The organizers

Mike Giles
Arnaud Doucet
Alexander Keller

University of Oxford Organizing Committee

Mike Giles	United Kingdom, Oxford
Arnaud Doucet	United Kingdom, Oxford
Abdul-Lateef Haji-Ali	United Kingdom, Heriot-Watt
Jeremy Heng	Singapore, ESSEC
Daniel Paulin	United Kingdom, Edinburgh
Alex Shestopaloff	United Kingdom, Alan Turing Institute
Lukasz Szpruch	United Kingdom, Edinburgh
Aretha Teckentrup	United Kingdom, Edinburgh

Berlin Organizing Committee

Alexander Keller	Germany, NVIDIA
Nikolaus Binder	Germany, NVIDIA
Thomas Müller	Germany, NVIDIA
Merlin Nimier-David	Switzerland, EPFL

Scientific Committee

Christophe Andrieu	United Kingdom, Bristol
Andrea Barth	Germany, Stuttgart
Zdravko Botev	Australia, University of New South Wales
Hector Cancela	Uruguay, University of the Republic
Frédéric Cérou	France, Inria
Nicolas Chopin	France, ENSAE
Ronald Cools	Belgium, KU Leuven
Josef Dick	Australia, University of New South Wales
Arnaud Doucet	United Kingdom, Oxford
Stefan Geiss	Finland, Jyväskylä
Mike Giles	United Kingdom, Oxford
Mark Girolami	United Kingdom, University of Warwick
Paul Glasserman	USA, Columbia
Michael Gnewuch	Germany, Osnabrück
Emmanuel Gobbet	France, Ecole Polytechnique
Takashi Goda	Japan, The University of Tokyo
Arnaud Guyader	France, University Pierre et Marie Curie
Stefan Heinrich	Germany, University Kaiserslautern
Fred J. Hickernell	USA, IIT
Aicke Hinrichs	Austria, JKU Linz

Wenzel Jakob	Switzerland, EPFL
Bert Kappen	Netherlands, Radboud
Alexander Keller	Germany, NVIDIA
Dirk Kroese	Australia, University Queensland
Frances Kuo	Australia, University of New South Wales
Gerhard Larcher	Austria, JKU Linz
Christian Lécot	France, University Savoie
Pierre L'Ecuyer	Canada, University Montréal
Christiane Lemieux	Canada, Waterloo
Gunther Leobacher	Austria, Graz
Faming Liang	USA, University Florida, Gainesville
Makoto Matsumoto	Japan, Hiroshima
Eric Moulines	France, Ecole Polytechnique
Thomas Müller-Gronbach	Germany, Passau
Andreas Neuenkirch	Germany, Mannheim
Harald Niederreiter	Austria, Academy of Sciences
Erich Novak	Germany, FSU, Jena
Dirk Nuyens	Belgium, KU Leuven
Art Owen	USA, Stanford University
Gilles Pagès	France, UPMC
Gareth Peters	United Kingdom, University College London
Friedrich Pillichshammer	Austria, JKU Linz
Mike Pitt	United Kingdom, KCL
Sebastian Reich	Germany, Potsdam
Klaus Ritter	Germany, Kaiserslautern
Gerardo Rubino	France, Inria
Wolfgang Schmid	Austria, Salzburg
Ian Sloan	Australia, University of New South Wales
Lukasz Szpruch	United Kingdom, Edinburgh
Aretha Teckentrup	United Kingdom, Edinburgh
Raul Tempone	Germany, Aachen
Bruno Tuffin	France, INRIA
Grzegorz Wasilkowski	USA, University Kentucky
Henryk Woźniakowski	USA, Columbia University

The MCQMC Conference Series

The MCQMC conference series is a biennial meeting focused on Monte Carlo (MC) and quasi-Monte Carlo (QMC) methods in scientific computing. Its aim is to provide a forum where leading researchers and users can exchange information on the latest theoretical developments and important applications of these methods. The events are held in alternate years with the International Conference on Monte Carlo Methods and Applications (MCM).

In a nutshell, Monte Carlo methods study complex systems by simulations fed by computer-generated pseudorandom numbers. Quasi-Monte Carlo methods replace these random numbers by more uniformly distributed and carefully selected numbers to improve their effectiveness. A large variety of special techniques have been developed and used to make these methods more effective in terms of speed and accuracy. The conference series focuses on the mathematical study of these techniques, their implementation and concrete applications, and their empirical assessment.

The conference was initiated by Harald Niederreiter [6], who co-chaired the first seven conferences. In 2006, Harald Niederreiter announced his wish to step down from the organizational role, and a steering committee was formed to ensure and oversee the continuation of the conference series. Both the steering committee and the locations of the so far 15 conferences are set out below.

If you are interested in hosting a future MCQMC conference at your institution, then please contact any member of the steering committee.

Steering Committee

Alexander Keller (Chair)	Germany, NVIDIA
Josef Dick	Australia, University of New South Wales
Fred J. Hickernell	USA, Illinois Institute of Technology
Pierre L'Ecuyer	Canada, University Montréal
Christiane Lemieux	Canada, University of Waterloo
Art Owen	USA, Stanford University
Friedrich Pillichshammer	Austria, Johannes Kepler University Linz

Conferences

We express our gratitude to Springer-Verlag for publishing the proceedings of the MCQMC conferences.

1. University of Nevada in Las Vegas, Nevada, USA, June 1994 [9]
2. University of Salzburg, Austria, July 1996 [8]
3. Claremont Colleges in Claremont, California, USA, June 1998 [10]
4. Hong Kong Baptist University in Hong Kong, China, November 2000 [4]
5. National University of Singapore, Republic of Singapore, November 2002 [7]
6. Palais des Congrès in Juan-les-Pins, France, June 2004 [11]
7. Ulm University, Germany, July 2006 [5]
8. Université de Montréal, Canada, July 2008 [3]
9. University of Warsaw, Poland, August 2010 [13]
10. University of New South Wales, Sydney, Australia, February 2012 [2]
11. Katholieke Universiteit Leuven, Belgium, April 2014 [1]
12. Stanford University, USA, August 2016 [12]
13. University of Rennes, France, July 2018 [14]
14. University of Oxford, United Kingdom, held online, August 2020
15. Johannes Kepler University in Linz, Austria, July 2022

References

1. Cools, R., Nuyens, D. (eds.): Monte Carlo and Quasi-Monte Carlo Methods, MCQMC, Leuven, Belgium, April 2014. Springer Proceedings in Mathematics & Statistics. Springer (2016). doi:https://doi.org/10.1007/978-3-319-33507-0
2. Dick, J., Kuo, F.Y., Peters, G.W., Sloan, I.H. (eds.): Monte Carlo and Quasi-Monte Carlo Methods 2012. Springer Proceedings in Mathematics & Statistics. Springer (2013). doi:https://doi.org/10.1007/978-3-642-41095-6
3. Ecuyer, P.L., Owen, A.B. (eds.): Monte Carlo and Quasi-Monte Carlo Methods 2008. Springer Proceedings in Mathematics & Statistics. Springer (2009). doi:https://doi.org/10.1007/978-3-642-04107-5
4. Fang, K.T., Hickernell, F.J., Niederreiter, H. (eds.): Monte Carlo and Quasi-Monte Carlo Methods 2000 Proceedings of a Conference held at Hong Kong Baptist University, Hong Kong SAR, China, November 27 – December 1, 2000. Springer Proceedings in Mathematics & Statistics. Springer (2002). doi:https://doi.org/10.1007/978-3-642-56046-0
5. Keller, A., Heinrich, S., Niederreiter, H. (eds.): Monte Carlo and Quasi-Monte Carlo Methods 2006. Springer Proceedings in Mathematics & Statistics. Springer (2008). doi:https://doi.org/10.1007/978-3-540-74496-2
6. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. SIAM, Philadelphia (1992)
7. Niederreiter, H. (ed.): Monte Carlo and Quasi-Monte Carlo Methods 2002, Proceedings of a Conference held at the National University of Singapore, Republic of Singapore, November 25–28, 2002. Springer Proceedings in Mathematics & Statistics. Springer (2004). doi:https://doi.org/10.1007/978-3-642-18743-8
8. Niederreiter, H., Hellekalek, P., Larcher, G., Zinterhof, P. (eds.): Monte Carlo and Quasi-Monte Carlo Methods 1996, Proceedings of a conference at the University of Salzburg, Austria, July 9–12, 1996. Springer Proceedings in Mathematics & Statistics. Springer (1998). doi:https://doi.org/10.1007/978-1-4612-1690-2

9. Niederreiter, H., Shiue, P.J. (eds.): Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Proceedings of a conference at the University of Nevada, Las Vegas, Nevada, USA, June 23–25, 1994. Springer Proceedings in Mathematics & Statistics. Springer (1995). doi:https://doi.org/10.1007/978-1-4612-2552-2
10. Niederreiter, H., Spanier, J. (eds.): Monte Carlo and Quasi-Monte Carlo Methods, Proceedings of a Conference held at the Claremont Graduate University, Claremont, California, USA, June 22–26, 1998. Springer Proceedings in Mathematics & Statistics. Springer (2000). doi:https://doi.org/10.1007/978-3-642-59657-5
11. Niederreiter, H., Talay, D. (eds.): Monte Carlo and Quasi-Monte Carlo Methods 2004. Springer Proceedings in Mathematics & Statistics. Springer (2006). doi:https://doi.org/10.1007/3-540-31186-6
12. Owen, A.B., Glynn, P.W. (eds.): Monte Carlo and Quasi-Monte Carlo Methods, MCQMC 2016, Stanford, CA, August 14-19. Springer Proceedings in Mathematics & Statistics. Springer (2018). doi:https://doi.org/10.1007/978-3-319-91436-7
13. Plaskota, L., Woźniakowski, H. (eds.): Monte Carlo and Quasi-Monte Carlo Methods 2010. Springer Proceedings in Mathematics & Statistics. Springer (2012). doi:https://doi.org/10.1007/978-3-642-27440-4
14. Tuffin, B., L'Ecuyer, P. (eds.): Monte Carlo and Quasi-Monte Carlo Methods, MCQMC 2018, Rennes, France, July 1–6. Springer Proceedings in Mathematics & Statistics. Springer (2020). doi:https://doi.org/10.1007/978-3-030-43465-6

Contents

Invited Talks and Tutorials

Density Estimation by Monte Carlo and Quasi-Monte Carlo

Pierre L'Ecuyer and Florian Puchhammer

Abstract Estimating the density of a continuous random variable X has been studied extensively in statistics, in the setting where n independent observations of X are given a priori and one wishes to estimate the density from that. Popular methods include histograms and kernel density estimators. In this review paper, we are interested instead in the situation where the observations are generated by Monte Carlo simulation from a model. Then, one can take advantage of variance reduction methods such as stratification, conditional Monte Carlo, and randomized quasi-Monte Carlo (RQMC), and obtain a more accurate density estimator than with standard Monte Carlo for a given computing budget. We discuss several ways of doing this, proposed in recent papers, with a focus on methods that exploit RQMC. A first idea is to directly combine RQMC with a standard kernel density estimator. Another one is to adapt a simulation-based derivative estimation method such as smoothed perturbation analysis or the likelihood ratio method to obtain a continuous estimator of the cumulative density function (CDF), whose derivative is an unbiased estimator of the density. This can then be combined with RQMC. We summarize recent theoretical results with these approaches and give numerical illustrations of how they improve the convergence of the mean square integrated error.

Keywords Density estimation · Conditional Monte Carlo · Likelihood ratio · Kernel density

P. L'Ecuyer (✉)
Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Montreal, Quebec, Canada
e-mail: lecuyer@iro.umontreal.ca

F. Puchhammer
Université de Montréal, Montreal, Quebec, Canada
e-mail: fpuchhammer@bcamath.org

Basque Center for Applied Mathematics, Bilbao, Spain

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_1

1 Introduction

In September 2015, the first author (PL) had an interesting lunchtime discussion with Art Owen and Fred Hickernell at a workshop on High-Dimensional Numerical Problems, at the Banff International Research Center, in the Canadian Rocky Mountains. It went as follows. In the MCQMC community, we focus largely on studying QMC and RQMC methods to estimate integrals that represent the mathematical expectations of certain random variables. In applications, the output random variable X of interest often represents a random cost or performance measure. But why estimate only the mean (the expectation) $\mathbb{E}[X]$? Data from simulation experiments can provide much more useful information than just an estimator and a confidence interval for $\mathbb{E}[X]$. When the number n of realizations of X is large enough, it permits one to estimate the entire distribution of X. And when X is a continuous random variable, this distribution is best visualized by showing its density. On the other hand, density estimation from a sample of n independent realizations of X is known to be a difficult problem in statistics. The leading density estimation methods, e.g., kernel density estimators (KDEs), only achieve a convergence rate of $O(n^{-4/5})$ for the mean square error (MSE) on the density at a given point, compared to a $O(n^{-1})$ rate for the expectation with MC. The main question raised in our 2015 discussion was: We know that RQMC can improve the $O(n^{-1})$ rate for the mean, but can it also improve the $O(n^{-4/5})$ rate for the density, by how much, and how?

Of course, this question makes sense only when the samples of X are obtained by simulation from a model, and not in the situation where n independent observations of X are given a priori. When the observations are generated from a model, there is room to change the way we generate them and construct the estimator, and in particular we may use RQMC points in place of independent uniform random numbers to generate the observations of X. Following this discussion, PL started exploring empirically what happens when we do this with an ordinary KDE. That is, what happens with the variance and MSE of the KDE estimator when the n observations of X are generated by simulation using a set of n RQMC points in place of n independent points, just like we do when estimating the mean. After much experiments and theoretical work with co-authors, this led to [3]. In that paper, we were able to prove an upper bound for the MSE with KDE+RQMC, but this bound converges at a faster rate than $O(n^{-4/5})$ only when the dimension s is very small. For moderate and large s, the bound converges at a slower rate than for crude Monte Carlo (MC), although the observed MSE was never larger than for MC in our experiments. The reason for the slow rate for the bound is that when increasing n, we need to reduce the bandwidth of the KDE to reduce the square bias and the MSE, but reducing the bandwidth increases rapidly the variation of the estimator as a function of the uniform random numbers, and this hurts the RQMC estimator.

We understood that for RQMC to be effective, we need smoother density estimators. In January 2017, while PL was visiting A. Owen at Stanford University to work on [3] he attended a talk by S. Asmussen who (by pure coincidence) was presenting [1], in which he shows how to obtain an unbiased density estimator for a sum

of independent random variables by conditional Monte Carlo. The conditioning is done by hiding the last variable in the sum and taking the density of the last variable right-shifted by the sum of other variables as a density estimator. We extended this idea to more general simulation models and this gave us what we needed to obtain smooth unbiased and RQMC-friendly density estimators. This led to the conditional density estimator (CDE) studied in [17], also presented in 2018 at a SAMSI workshop on QMC methods in North Carolina and at a RICAM workshop in Austria. The idea of this CDE method is to define a continuous estimator of the CDF $F(x)$ by conditioning, and take its sample derivative with respect to x as a density estimator. Under appropriate conditions, this provides an unbiased density estimator, and when further favorable conditions hold, this estimator can be smooth and RQMC-friendly. In March 2021, while we were finalizing this paper, Mike Fu pointed out that [6] already contains an example in which he uses conditional Monte Carlo to estimate the density of the length of the longest path in a six-link network in which the last link is shared by all paths. His unbiased density estimator is essentially the same as in [1]: it is the density of the length of the last link, right-shifted by the length of the longest path up to that link.

At the Eleventh International Conference on Monte Carlo Methods and Applications (MCM), in July 2017, the authors of [10] presented a different approach that can provide an unbiased density estimator for a sum of random variables as in [1], except that the variables can be dependent. This approach can be generalized to obtain a continuous CDF estimator and then an unbiased density estimator, via the likelihood ratio (LR) simulation-based derivative estimation method [7, 11] and a clever change of variable, and by taking again the sample derivative of this CDF estimator. This likelihood ratio density estimator (LRDE) is discussed in Sect. 6 and also in [16]. We also explain how it can be combined with RQMC.

A generalized version of the LR gradient estimator method, named GLR, was proposed in [23] to handle situations in which neither the usual LR estimator nor the direct sample derivative apply, because of discontinuities. In [18], the authors sketch out how this GLR method could be used to obtain an unbiased density estimator. Their general formulas are not easy to understand and implement, but more convenient formulas for these GLR density estimators are given in Theorem 1 of [22]. A modified version of the GLR named GLR-U was developed recently in [24] to handle large classes of situations that could not be handled easily by the original GLR from [23]. The model of [24] is expressed explicitly in terms of independent uniform random variables over $(0, 1)$. Density estimators can also be obtained by this method.

All these LR and GLR methods use a multivariate change of variable of some sort. They provide unbiased density estimators that are often not smooth with respect to the underlying uniforms, so their direct combination with RQMC does not always bring much gain. However, it is often possible to smooth out the LR, GLR, or GLR-U density estimator by conditioning just before applying RQMC.

The aim of this paper is to provide an overview of these recent developments on density estimation for simulation models, by MC and RQMC. We summarize the main theoretical results and give numerical illustrations on how the estimators behave, using simple examples.

The remainder is organized as follows. In Sects. 2 and 3, we recall basic facts about one-dimensional density estimation and RQMC sampling. In Sect. 4, we summarize what happens when we directly combine a KDE with RQMC. In Sect. 5, we discuss the CDE and its combination with RQMC. In Sect. 6, we examine the LR and GLR density estimators. Section 8 gives numerical illustrations. We wrap up with a conclusion in Sect. 9.

2 Basic Density Estimation

Let X be a continuous real-valued random variable with CDF F and density f. The goal is to estimate the density f over a finite interval $[a, b]$, from a sample $X_1, \ldots, X_n$ of n realizations of X (not necessarily independent). This problem has been studied at length in statistics for the case where $X_1, \ldots, X_n$ are independent [26]. To measure the quality of an arbitrary density estimator $\hat{f}_n$ based on this sample, we will use the *mean integrated square error* (MISE), which is the integral of the MSE over the interval $[a, b]$:

$$\text{MISE} = \text{MISE}(\hat{f}_n) = \int_a^b \mathbb{E}[\hat{f}_n(x) - f(x)]^2 \mathrm{d}x = \text{IV} + \text{ISB} \tag{1}$$

where

$$\text{IV} = \int_a^b \mathbb{E}(\hat{f}_n(x) - \mathbb{E}[\hat{f}_n(x)])^2 \mathrm{d}x \quad \text{and} \quad \text{ISB} = \int_a^b (\mathbb{E}[\hat{f}_n(x)] - f(x))^2 \mathrm{d}x$$

are the *integrated variance* (IV) and the *integrated square bias* (ISB), respectively.

Two popular types of density estimators are histograms and KDEs. To define a *histogram*, one can partition $[a, b]$ into m intervals of length $h = (b - a)/m$ and put

$$\hat{f}_n(x) = \frac{n_j}{nh} \quad \text{for } x \in I_j = [a + (j-1)h, a + jh), \quad j = 1, \ldots, m,$$

where n_j is the number of observations X_i that fall in interval I_j. To define a KDE [21, 26], select a *kernel* k (usually a unimodal symmetric density centered at 0) and a *bandwidth* $h > 0$ (an horizontal stretching factor for the kernel), and put

$$\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} k\left(\frac{x - X_i}{h}\right).$$

These two density estimators are biased. Asymptotically, when $n \to \infty$ and $h \to 0$ jointly, in the case of independent samples $X_1, \ldots, X_n$, the IV and ISB behave as

Table 1 Constants involved in the convergence rates of the MISE for histograms and KDEs

	C	B	α	h^*	MISE
Histogram	1	$R(f')/12$	2	$(nR(f')/6)^{-1/3}$	$O(n^{-2/3})$
KDE	$\mu_0(k^2)$	$(\mu_2(k))^2 R(f'')/4$	4	$\left(\frac{\mu_0(k^2)}{(\mu_2(k))^2 R(f'')n}\right)^{1/5}$	$O(n^{-4/5})$

$$\text{MISE} \;=\; \text{IV} + \text{ISB} \;\sim\; C/(nh) + Bh^{\alpha}$$

where C, B, and α depend on the method. The asymptotically optimal h is then

$$h^* = (C/(B\alpha n))^{1/(\alpha+1)}$$

and it gives MISE $\sim K n^{-\alpha/(1+\alpha)}$ for some constant K. Table 1 gives expressions for C, B, α, h^*, and $\alpha/(1+\alpha)$, for histograms and KDEs, with independent samples. It uses the following definitions, for any $g : \mathbb{R} \to \mathbb{R}$:

$$R(g) = \int_a^b (g(x))^2 \mathrm{d}x \quad \text{and} \quad \mu_r(g) = \int_{-\infty}^{\infty} x^r g(x) \mathrm{d}x \quad \text{for } r = 0 \text{ and } 2.$$

Note that these expressions hold under the simplifying assumption that h must be the same all over $[a, b]$. One may often do better by varying the bandwidth over $[a, b]$, but this is more complicated. To estimate h^* in practice, one can estimate $R(f')$ and $R(f'')$ by using a KDE to estimate f' and f'' (very roughly). This type of crude (plugin) estimate is often good enough. In the following, we will see how to improve on these MISE rates and values in a simulation setting, by reducing the variance. In general, using RQMC points instead of MC does not change the bias.

3 RQMC

We recall here some basic principles of RQMC used in the forthcoming sections. For more extensive coverages, see [4, 12, 13, 20], for example. Suppose we want to estimate $\mu = \mathbb{E}[g(\boldsymbol{U})]$ where $\boldsymbol{U} = (U_1, \ldots, U_s)$ has the uniform distribution over the s-dimensional unit cube $(0, 1)^s$ and $g : (0, 1)^s \to \mathbb{R}$. With standard MC, we draw n independent random points $\boldsymbol{U}_i$ uniformly over $(0, 1)^s$ and we estimate the expectation by the average

$$\hat{\mu}_{n,\text{mc}} = \frac{1}{n} \sum_{i=1}^{n} g(\boldsymbol{U}_i). \tag{2}$$

With RQMC, we replace the independent random points $\boldsymbol{U}_i$ by a set of *dependent* random points $\tilde{P}_n = \{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n\} \subset (0, 1)^s$ such that (1) the point set $\tilde{P}_n$ covers

the unit hypercube very evenly (in a sense that must be precisely defined) with probability 1; and (2) each point $\boldsymbol{U}_i$ has the uniform distribution over $(0, 1)^s$. Then we estimate the expectation by the same average as in (2), which we now denote $\hat{\mu}_{n,\text{rqmc}}$. For various spaces $\mathcal{H}$ of functions g, usually Hilbert or Banach spaces, we have inequalities of the form

$$\text{Var}[\hat{\mu}_{n,\text{rqmc}}] \le \mathcal{D}^2(P_n) \cdot \mathcal{V}^2(g) \tag{3}$$

where $\mathcal{D}(P_n)$ measures the *discrepancy* of P_n (with respect to the uniform distribution) and $\mathcal{V}(g)$ measures the variation of the function g. For many of these function spaces, we also know explicitly how to construct RQMC point sets for which $\mathcal{D}(P_n) = O(n^{-\alpha/2}(\log n)^{s-1})$ for some $\alpha > 1$ [4, 9, 14]. This leads to

$$\text{Var}[\hat{\mu}_{n,\text{rqmc}}] = O(n^{-\alpha}(\log n)^{2(s-1)})$$

when $\mathcal{V}(f) < \infty$. A classical case is the standard Koksma-Hlawka inequality, for which $\alpha = 2$, $\mathcal{D}(P_n) = \mathcal{D}^*(P_n)$ is the star discrepancy, and $\mathcal{V}(g)$ is the variation in the sense of Hardy and Krause, defined by

$$\mathcal{V}(g) = \mathcal{V}_{\text{HK}}(g) = \sum_{\emptyset \ne \mathfrak{v} \subseteq \{1,\ldots,s\}} \int_{(0,1)^{|\mathfrak{v}|}} \left| \frac{\partial^{|\mathfrak{v}|}}{\partial \boldsymbol{u}_{\mathfrak{v}}} g(\boldsymbol{u}_{\mathfrak{v}}, \mathbf{1}) \right| \mathrm{d}\boldsymbol{u}_{\mathfrak{v}}, \tag{4}$$

where $\boldsymbol{u}_{\mathfrak{v}}$ is the vector of coordinates whose indices belong to $\mathfrak{v}$, $|\mathfrak{v}|$ is the cardinality of $\mathfrak{v}$, and under the assumption that this expression is well defined. The main construction methods for P_n are lattice rules and digital nets.

In the context of density estimation, the average in (2) is replaced by the density estimator $\hat{f}_n(x)$ at a given point x. If our density estimator can be written as an average of the form

$$\hat{f}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \tilde{g}(x, \boldsymbol{U}_i) \tag{5}$$

where $\tilde{g}$ is a sufficiently smooth function of its second argument, then we can apply the RQMC theory just described to this density estimator by replacing the function $g(\cdot)$ by $\tilde{g}(x, \cdot)$. We look at this in the next few sections.

4 Kernel Density Estimators with RQMC

The KDE at a given point $x \in [a, b]$ is

$$\hat{f}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} k\left(\frac{x - g(\boldsymbol{U}_i)}{h}\right) = \frac{1}{n} \sum_{i=1}^{n} \tilde{g}(x, \boldsymbol{U}_i).$$

We assume that the kernel k is a smooth probability density, symmetric about 0, and at least s times differentiable everywhere. With RQMC points $\boldsymbol{U}_i$, this is an RQMC estimator of $\mathbb{E}[\tilde{g}(x, \boldsymbol{U})] = \mathbb{E}[\hat{f}_n(x)]$. RQMC does not change the bias of this density estimator, but it may reduce $\mathrm{Var}[\hat{f}_n(x)]$, which would reduce in turn the IV and the MISE.

To prove RQMC variance bounds via (3), we need to bound the variation $\mathcal{V}(\tilde{g})$. This was done in [3] for the classical Hardy-Krause variation (4), which is bounded if and only if all the partial derivatives

$$\frac{\partial^{|\mathfrak{v}|}}{\partial \boldsymbol{u}_{\mathfrak{v}}} \tilde{g}(x, \boldsymbol{u}) = \frac{1}{h} \frac{\partial^{|\mathfrak{v}|}}{\partial \boldsymbol{u}_{\mathfrak{v}}} k\left(\frac{x - g(\boldsymbol{u})}{h}\right)$$

exist and are uniformly bounded. The derivatives with respect to k are easily bounded for instance if k is a standard normal density (the Gaussian kernel). However, when expanding the derivatives via the chain rule, we obtain terms in h^{-j} for $j = 2, \ldots, |\mathfrak{v}| + 1$. The dominant term asymptotically is the term for $|\mathfrak{v}| = s$, and it grows in general as $h^{-s-1} \left| k^{(s)}\left((x - g(\boldsymbol{u}))/h\right) \prod_{j=1}^{s} g_{\{j\}}(\boldsymbol{u}) \right| = O(h^{-s-1})$ when $h \to 0$, where $g_{\{j\}}$ is the derivative of g with respect to its jth coordinate. We can bring it down to $O(h^{-s})$ via a change of variables, which leads to the following result proved in [3]:

Proposition 1 *Let* $g : [0, 1]^s \to \mathbb{R}$ *be* piecewise monotone *in each coordinate* u_j *when the other coordinates are fixed. Assume that all first-order partial derivatives of* g *are continuous and that* $\|g_{\mathfrak{w}_1} g_{\mathfrak{w}_2} \cdots g_{\mathfrak{w}_\ell}\|_1 < \infty$ *for all selections of non-empty, mutually disjoint index sets* $\mathfrak{w}_1, \ldots, \mathfrak{w}_\ell \subseteq \{1, \ldots, s\}$*, where* $g_{\mathfrak{w}}$ *is the derivative of* g *with respect to all the coordinates in the index set* $\mathfrak{w}$.

Then the Hardy-Krause variation of $\tilde{g}(x, \cdot)$ *for any fixed* $x \in [a, b]$ *satisfies*

$$\mathcal{V}_{\mathrm{HK}}(\tilde{g}(x, \cdot)) \le c h^{-s} + O(h^{-s+1})$$

for some constant $c > 0$ *given in [3], and with RQMC point sets having a star discrepancy* $\mathcal{D}^*(P_n) = O(n^{-1+\epsilon})$ *for all* $\epsilon > 0$ *when* $n \to \infty$*, we obtain*

$$\mathrm{IV} = O(n^{-2+\epsilon} h^{-2s}) \quad \textit{for all } \epsilon > 0.$$

RQMC does not change the bias, so the ISB has exactly the same expression as for MC. By picking h *to minimize the MISE bound, we get* $\mathrm{MISE} = O(n^{-4/(2+s)+\epsilon})$.

This rate for the MISE is worse than the MC rate when $s \ge 4$. The factor h^{-2s} in the IV bound really hurts. On the other hand, this is only an upper bound, not the actual IV. Proposition 4.4 of [3] also shows via a different analysis that for the KDE, there exist RQMC constructions for which the asymptotic decrease rate of the IV is not worse than for MC.

5 Conditional Density Estimation with RQMC

To estimate the density $f(x) = F'(x)$, one may think of simply taking the sample derivative of an unbiased estimator of the CDF $F(x)$. The simplest unbiased estimator of this CDF is the *empirical CDF*

$$\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[X_i \le x].$$

However $\mathrm{d}\hat{F}_n(x)/\mathrm{d}x = 0$ almost everywhere, so this *cannot* be a useful density estimator! We need a smoother CDF estimator, which should be at least continuous in x.

One effective way of smoothing an estimator and often make it continuous is to replace it by its conditional expectation given partial (filtered) information. This is *conditional Monte Carlo* (CMC) [2]. That is, one replaces the indicator $\mathbb{I}[X_i \le x]$ in the expression of $\hat{F}_n(x)$ above by the conditional CDF $F(x \mid \mathcal{G}) = \mathbb{P}[X_i \le x \mid \mathcal{G}]$, where $\mathcal{G}$ is a sigma-field that contains not enough information to reveal X but enough to compute $F(x \mid \mathcal{G})$, then one takes the sample derivative. We call it the *conditional density estimator* (CDE). For more details about the CMC method in general and the choice of $\mathcal{G}$ in specific cases, see for example [2, 5, 15]. For examples in the context of density estimation, see [17] and the examples in Sect. 8. We assume here that we can compute the conditional density either directly or numerically by an iterative algorithm. The following proposition, proved in [17], gives sufficient conditions for this CDE to be an unbiased density estimator with finite variance.

Proposition 2 *Suppose that for all realizations of* $\mathcal{G}$, $F(x \mid \mathcal{G})$ *is a* continuous *function of* x *over* $[a, b]$, differentiable *except perhaps over a denumerable set of points* $D(\mathcal{G}) \subset [a, b]$, *and for which* $f(x \mid \mathcal{G}) = F'(x \mid \mathcal{G}) = \mathrm{d}F(x \mid \mathcal{G})/\mathrm{d}x$ *(when it exists) is bounded uniformly in* x *by a random variable* Γ *such that* $\mathbb{E}[\Gamma^2] \le K_\gamma < \infty$. *Then, for all* $x \in [a, b]$, $\mathbb{E}[f(x \mid \mathcal{G})] = f(x)$ *and* $\mathrm{Var}[f(x \mid \mathcal{G})] < K_\gamma$. *Moreover, if* $\mathcal{G} \subset \tilde{\mathcal{G}}$ *both satisfy the assumptions of this proposition, then* $\mathrm{Var}[f(x \mid \mathcal{G})] \le \mathrm{Var}[f(x \mid \tilde{\mathcal{G}})]$.

For a sample of size n, the CDE becomes

$$\hat{f}_{\mathrm{cde},n}(x) = \frac{1}{n}\sum_{i=1}^{n} f(x \mid \mathcal{G}^{(i)})$$

where $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(n)}$ are n "realizations" of $\mathcal{G}$. When the n realizations are independent we have $\mathrm{Var}[\hat{f}_{\mathrm{cde},n}(x)] \le K_\gamma/n = O(n^{-1})$.

To combine the CDE with RQMC, we want to write $f(x \mid \mathcal{G}) = \tilde{g}(x, \boldsymbol{u})$ for some function $\tilde{g} : [a, b] \times [0, 1)^s \to \mathbb{R}$. This function $\tilde{g}(x, \cdot)$ will be used in (5). The combined CDE+RQMC estimator is then defined by

$$\hat{f}_{\text{cde-rqmc},n}(x) = \frac{1}{n}\sum_{i=1}^{n} \tilde{g}(x, \boldsymbol{U}_i). \tag{6}$$

where $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n\}$ is an RQMC point set.

If $\tilde{g}(x, \cdot)$ has *bounded variation*, then we can get an $O(n^{-2+\epsilon})$ *rate for the MISE*, and sometimes better. This holds in several examples that we tried. If $\tilde{g}(x, \cdot)$ has *unbounded variation*, RQMC may still reduce the IV, but there is no guarantee.

6 Likelihood Ratio Density Estimators

There are situations where a CDE as in Sect. 5 might be too difficult to obtain. An alternative can be a *likelihood ratio density estimator* (LRDE), defined as follows. Suppose that $X = h(\boldsymbol{Y})$ where $\boldsymbol{Y}$ has known density $f_{\boldsymbol{Y}}$ over $\mathbb{R}^s$, and we know how to generate it and compute $X = h(\boldsymbol{Y})$. For simplicity, let $x > 0$ (in case we are really interested in some $x \le 0$, we can simply add a constant to the function h). We have

$$F(x) = \mathbb{P}[h(\boldsymbol{Y}) \le x] = \int_{\mathbb{R}^s} \mathbb{I}[h(\boldsymbol{y})/x \le 1] f_{\boldsymbol{Y}}(\boldsymbol{y}) \mathrm{d}\boldsymbol{y}.$$

We want to change this integrand into a continuous function of x, so we can take the derivative with respect to x inside the integral. One way to do this is to make a change of variable $\boldsymbol{y} \mapsto \boldsymbol{z} = \boldsymbol{z}(x)$ of the form $\boldsymbol{y} = \varphi_x(\boldsymbol{z})$, with Jacobian $|J_x(\boldsymbol{z})|$, so that $\tilde{h}(\boldsymbol{z}) = h(\varphi_x(\boldsymbol{z}))/x$ no longer depends on x for any given $\boldsymbol{z}$. We can then rewrite

$$F(x) = \int_{\mathbb{R}^s} \mathbb{I}[\tilde{h}(\boldsymbol{z}) \le 1] f_{\boldsymbol{Y}}(\varphi_x(\boldsymbol{z}))|J_x(\boldsymbol{z})| \mathrm{d}\boldsymbol{z}.$$

In a small open neighborhood of a given $x_0 \in [a, b]$, we have

$$F(x) = \int_{\mathbb{R}^s} \mathbb{I}[\tilde{h}(\boldsymbol{z}) \le 1] L(\boldsymbol{z}; x, x_0) f_{\boldsymbol{Y}}(\varphi_{x_0}(\boldsymbol{z}))|J_{x_0}(\boldsymbol{z})| \mathrm{d}\boldsymbol{z}$$

where

$$L(\boldsymbol{z}; x, x_0) = \frac{f_{\boldsymbol{Y}}(\varphi_x(\boldsymbol{z}))|J_x(\boldsymbol{z})|}{f_{\boldsymbol{Y}}(\varphi_{x_0}(\boldsymbol{z}))|J_{x_0}(\boldsymbol{z})|}$$

is the *likelihood ratio* between the density of $\boldsymbol{z}$ at x and at x_0. Under appropriate conditions:

$$
\begin{aligned}
f(x) &= \frac{\mathrm{d}}{\mathrm{d}x} \int_{\mathbb{R}^s} \mathbb{I}[\tilde{h}(\boldsymbol{z}) \le 1] L(\boldsymbol{z}; x, x_0) f_{\boldsymbol{Y}}(\varphi_{x_0}(\boldsymbol{z})) |J_{x_0}(\boldsymbol{z})| \mathrm{d}\boldsymbol{z} \\
&= \int_{\mathbb{R}^d} \mathbb{I}[\tilde{h}(\boldsymbol{z}) \le 1] \left(\frac{\mathrm{d}}{\mathrm{d}x} L(\boldsymbol{z}; x, x_0) \right) \frac{f_{\boldsymbol{Y}}(\varphi_x(\boldsymbol{z})) |J_x(\boldsymbol{z})|}{L(\boldsymbol{z}; x, x_0)} \mathrm{d}\boldsymbol{z} \\
&= \int_{\mathbb{R}^d} \mathbb{I}[\tilde{h}(\boldsymbol{z}) \le 1] \left(\frac{\mathrm{d}}{\mathrm{d}x} \ln L(\boldsymbol{z}; x, x_0) \right) f_{\boldsymbol{Y}}(\varphi_x(\boldsymbol{z})) |J_x(\boldsymbol{z})| \mathrm{d}\boldsymbol{z} \\
&= \int_{\mathbb{R}^s} \mathbb{I}[h(\boldsymbol{y}) \le x] S(\boldsymbol{y}, x) f_{\boldsymbol{Y}}(\boldsymbol{y}) \mathrm{d}\boldsymbol{y}
\end{aligned}
$$

where

$$
S(\boldsymbol{y}, x) = \frac{\mathrm{d} \ln L(\boldsymbol{z}; x, x_0)}{\mathrm{d}x} = (\nabla(\ln f_{\boldsymbol{Y}})(\boldsymbol{y})) \cdot (\nabla_x \varphi_x(\boldsymbol{z})) + \frac{\mathrm{d} \ln |J_x(\boldsymbol{z})|}{\mathrm{d}x}
$$

is the *score function* associated with L. This gives the unbiased LRDE

$$
\hat{f}_{\mathrm{lrde}}(x) = \mathbb{I}[h(\boldsymbol{Y}) \le x] \, S(\boldsymbol{Y}, x) \tag{7}
$$

where $\boldsymbol{Y} \sim f_{\boldsymbol{Y}}$. Here, $\boldsymbol{Y}$ can have a multivariate distribution for which conditioning is hard whereas $S(\boldsymbol{Y}, x)$ may be easier to compute.

This LR approach has been widely used to estimate the derivative of $\mathbb{E}[h(\boldsymbol{Y})]$ with respect to a parameter of the distribution of $\boldsymbol{Y}$ [2, 7, 8, 11]. Laub et al. [10] obtained (via a different argument) the estimator (7) for the special case where $h(\boldsymbol{Y})$ is a sum of random variables. The following is proved in [16].

Proposition 3 *Suppose that with probability one over realizations of $\boldsymbol{Y} = \varphi_x(\boldsymbol{Z})$, $f_{\boldsymbol{Y}}(\varphi_x(\boldsymbol{Z}))|J_x(\boldsymbol{Z})|$ is continuous in x over $[a, b]$ and is differentiable in x except perhaps at a countable set of points $D(\boldsymbol{Y}) \subset [a, b]$. Suppose that there is also a random variable Γ defined over the same probability space as $\boldsymbol{Y}$, such that $\mathbb{E}[\Gamma^2] < \infty$, and for which*

$$
\sup_{x \in [a,b] \setminus D(\boldsymbol{Y})} |\mathbb{I}[h(\boldsymbol{Y}) \le x] \, S(\boldsymbol{Y}, x)| \le \Gamma.
$$

Then, $\hat{f}_{\mathrm{lrde}}(x) = \mathbb{I}[h(\boldsymbol{Y}) \le x] \, S(\boldsymbol{Y}, x)$ is an unbiased estimator of $f(x)$ at almost all $x \in [a, b]$, with variance bounded uniformly by $\mathbb{E}[\Gamma^2]$.

Note that the unbiased LRDE in (7) is usually discontinuous in the underlying uniforms, because of the indicator function, so it is not a smooth RQMC-friendly estimator. One can think of making it continuous by taking its conditional expectation. On the other hand, when we can find a conditioning that makes the indicator continuous, then we may be able to apply the CDE instead and this is usually more effective, according to our experiments. The LRDE is nevertheless useful for the situations in which a CDE is difficult to obtain.

7 Generalized Likelihood Ratio Estimators

Peng et al. [23] proposed a *generalized likelihood ratio* (GLR) method that generalizes the LR derivative estimation approach. Peng et al. [22] gave an adaptation of this method to density estimation. It goes as follows. Let $X = h(\boldsymbol{Y}) = h(Y_1, \ldots, Y_s)$ for some random variables $Y_1, \ldots, Y_s$, and assume that X is a continuous random variable with (unknown) density f. Let $A(x, \epsilon) = \{\boldsymbol{y} \in \mathbb{R}^s : x - \epsilon \le h(\boldsymbol{y}) \le x + \epsilon\}$, which is the inverse image of an ϵ-neighborhood of x by h. Suppose there is an $\epsilon_0 > 0$ such that

$$\lim_{\epsilon \to 0} \sup_{x \in [a-\epsilon_0, b+\epsilon_0]} \lambda(A(x, \epsilon)) = 0,$$

where λ is the Lebesgue measure on $\mathbb{R}^s$. Select some index $j \in \{1, \ldots, s\}$ for which Y_j is a continuous random variable with CDF F_j and density f_j, and is independent of $\{Y_k,\ k \ne j\}$. Let $h_{(j)}(\boldsymbol{y}) := \partial h(\boldsymbol{y})/\partial y_j$, $h_{(jj)}(\boldsymbol{y}) := \partial^2 h(\boldsymbol{y})/\partial y_j^2$, and

$$\Psi_j(\boldsymbol{y}) = \frac{\partial \ln f_j(y_j)/\partial y_j - h_{(jj)}(\boldsymbol{y})/h_{(j)}(\boldsymbol{y})}{h_{(j)}(\boldsymbol{y})},$$

where all these derivatives are assumed to exist. Suppose that there are functions $v_\ell : \mathbb{R} \to \mathbb{R}$ for $\ell = 1, \ldots, s$ such that $|h_{(j)}(\boldsymbol{y})|^{-1} \le \prod_{\ell=1}^d v_\ell(y_\ell)$ and

$$\lim_{y \to \pm\infty} v_j(y) f_j(y) = 0 \ \text{ and } \ \mathbb{E}[v_j(Y_j)] < \infty.$$

Finally, suppose also that $\mathbb{E}[\mathbb{I}[X \le x]\Psi_j^2(\boldsymbol{Y})] < \infty$. Under all these conditions, a simple modification of the proof of Theorem 1 in [22] yields the following:

Proposition 4 *With the assumptions just given, $D_j(x, \boldsymbol{Y}) = \mathbb{I}[X \le x]\Psi_j(\boldsymbol{Y})$ is an unbiased and finite-variance estimator of the density $f(x)$ at x.*

When the conditions hold for all $j = 1, \ldots, s$, as assumed in [22], this gives s unbiased estimators $D_1(x, \boldsymbol{Y}), \ldots, D_s(x, \boldsymbol{Y})$. Instead of selecting only one of them, we can take a linear combination $D(x, \boldsymbol{Y}) = w_1 D_1(x, \boldsymbol{Y}) + \cdots + w_s D_s(x, \boldsymbol{Y})$ where $w_1 + \cdots + w_s = 1$. This is exactly equivalent to taking, say $D_1(x, \boldsymbol{Y})$ as the base estimator and the $C_j = D_j(x, \boldsymbol{Y}) - D_1(x, \boldsymbol{Y})$ as mean-zero control variates, for $j = 2, \ldots, s$, because one has $D(x, \boldsymbol{Y}) = D_1(x, \boldsymbol{Y}) + w_2 C_2 \cdots + w_s C_s$. Therefore, standard control variate theory [2] can be used to optimize the coefficients w_j. When the conditions are satisfied only for certain values of j, then one can take the linear combination only for these values. It may also happen that the assumptions are satisfied for no j, in which case this method does not apply.

The GLR setting of [23] is more general. It permits one to estimate the derivative of $\mathbb{E}[\varphi(g(\boldsymbol{Y}; \theta))]$ with respect to some parameter θ, where $g(\cdot; \theta) : \mathbb{R}^s \to \mathbb{R}^s$ is continuous and one-to-one for the values of θ in the region of interest, so it corresponds to a multivariate change of variable in that region. The authors provide a general form of the unbiased derivative estimator (see also [25]). The general for-

mula is rather complicated and it can be found in the papers. One can use it in principle to estimate the density of X by taking $\theta = x$ and selecting a g for which $\varphi(g(\boldsymbol{Y}; \theta)) \equiv \mathbb{I}[X \le x] = \mathbb{I}[h(\boldsymbol{Y}) - x \le 0]$ and for which the assumptions of [23] are satisfied, when this is possible.

Peng et al. [24] extended the range of applicability of GLR by developing GLR-U, a version of GLR in which the base model is expressed directly in terms of the underlying uniform random numbers. That is, $\boldsymbol{Y}$ takes the form of a vector $\boldsymbol{U}$ which has the uniform distribution over the unit hypercube $(0, 1)^s$. This new setting covers a larger class of models than in [23], including situations where the random variables are generated by inversion, by the rejection method, or via Archimedean copulas, for example. We outline how to use this method to estimate the density of X over $[a, b]$.

The first step is to find a nonempty subset of the input variables $\Upsilon \subseteq \{1, \ldots, s\}$, which we will assume (without loss of generality) to be $\Upsilon = \{1, \ldots, d\}$ for $1 \le d \le s$, together with a function $g(\cdot; x) = g_1(\cdot; x), \ldots, g_d(\cdot; x)) : (0, 1)^s \to \mathbb{R}^d$ for which $\varphi(g(\boldsymbol{U}; x)) \equiv \mathbb{I}[X \le x]$ for all $x \in [a, b]$ and which satisfies the following assumptions. For any $\boldsymbol{u} \in (0, 1)^s$, we decompose $\boldsymbol{u} = (\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)})$ where $\boldsymbol{u}^{(1)}$ contains the first d coordinates and $\boldsymbol{u}^{(2)}$ the other ones. When $\boldsymbol{u}^{(2)}$ is fixed, $g(\cdot; x)$ becomes a function of $\boldsymbol{u}^{(1)}$ only, which we denote by $\tilde{g}(\cdot; \boldsymbol{u}^{(2)}, x)$. An important condition is that this function $\tilde{g}$ must be continuous and correspond to a multivariate change of variable, whose Jacobian $J_g(\boldsymbol{u}; x)$ is a $d \times d$ invertible matrix whose element (i, j) is $\partial g_i(\boldsymbol{u}; x)/\partial u_j$. For any $\boldsymbol{u} = (u_1, \ldots, u_s) \in (0, 1)^s$ and $j = 1, \ldots, d$, let $\overline{\boldsymbol{u}}_j$ and $\underline{\boldsymbol{u}}_j$ be the vector $\boldsymbol{u}$ in the limit when $u_j \to 1$ from the left and the limit when $u_j \to 0$ from the right (see [24, 25]). Define

$$r_j(\boldsymbol{u}; x) = -(J_g^{-1}(\boldsymbol{u}; x))^{\mathsf{t}} \cdot \boldsymbol{e}_j$$

and

$$v(\boldsymbol{u}; x) = -\sum_{j=1}^{d} \boldsymbol{e}_j^{\mathsf{t}} \cdot (J_g^{-1}(\boldsymbol{u}; x)) \left(\frac{\mathrm{d} J_g(\boldsymbol{u}; x)}{\mathrm{d} u_j} \right) (J_g^{-1}(\boldsymbol{u}; x)) \cdot \mathbf{1}$$

where $\boldsymbol{e}_j$ is the jth unit vector, $\mathbf{1}$ is a column vector of ones, and the derivative of $J_g(\boldsymbol{u}; x)$ is element-wise. Then, under some mild regularity conditions, we have:

Proposition 5 *The following is an unbiased density estimator at all $x \in [a, b]$:*

$$G(\boldsymbol{U}, x) = \mathbb{I}[X \le x]\, v(\boldsymbol{U}; x) + \sum_{j=1}^{s} \Big[\varphi(g(\overline{\boldsymbol{U}}_j; x)) r_j(\overline{\boldsymbol{U}}_j; x) - \varphi(g(\underline{\boldsymbol{U}}_j; x)) r_j(\underline{\boldsymbol{U}}_j; x) \Big]. \tag{8}$$

Peng et al. [24, 25] show how to apply this method in the special case where X is the maximum of several variables, each one being the sum of certain Y_j's that are generated by inversion from the U_j's. This may correspond to the length of the longest path between a source node to a destination node in a directed network, for

example. It works in the same way if the maximum is replaced by a minimum, and we will use it in Sect. 8. The number d of selected input variables in this case should be equal to the number of independent paths.

8 Numerical Illustrations

We illustrate the applicability and performance of the various density estimators discussed here on a small shortest path example defined below. We run the simulations with MC and RQMC. For RQMC, we use Sobol' nets with direction numbers taken from [19], and randomized by a left matrix scramble followed by a digital shift. Each RQMC experiment is repeated $m = 100$ times independently. The performance is assessed via the estimated MISE for $n = 2^{20}$ points. For RQMC, we also estimate the convergence rate as follows: we assume that MISE $\approx n^{-\beta}$ for some constant $\beta > 0$ and we estimate β by $\hat{\beta}$ using linear regression in log scale, based on observations obtained with $n = 2^{13}, 2^{14}, \ldots, 2^{20}$. For MC, the rates are known theoretically to be $\beta = 0.8$ for the KDE and $\beta = 1$ for the other methods. For the experiments with the KDE, we select the bandwidth with the same methodology as in [3].

We consider an acyclic directed network as in Fig. 1, with s arcs. For $j = 1, \ldots, s$, arc j has random length Y_j with continuous cdf F_j and density f_j, and the Y_j are assumed independent. We generate Y_j by inversion via $Y_j = F_j^{-1}(U_j)$ where $U_j \sim U(0, 1)$. We want to estimate the density of the length X of the shortest path from the source to the sink.

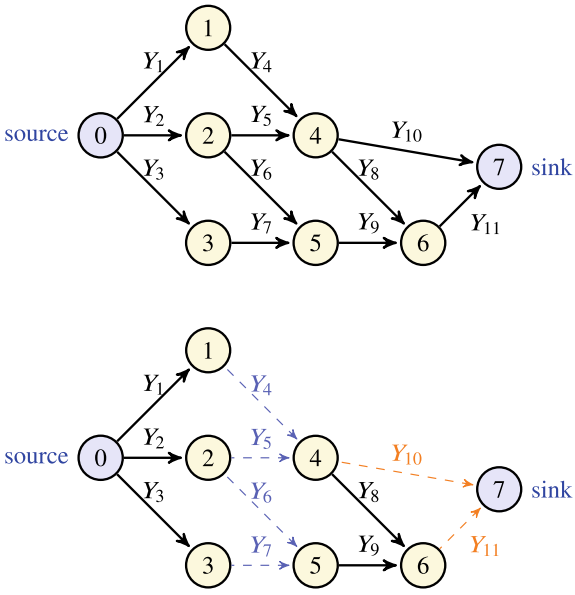


Fig. 1 Upper panel: a directed network with 11 links. Lower panel: two selected minimal cuts $\mathcal{L}_1 = \{4, 5, 6, 7\}$ (in light blue) and $\mathcal{L}_2 = \{10, 11\}$ (in orange) for this network

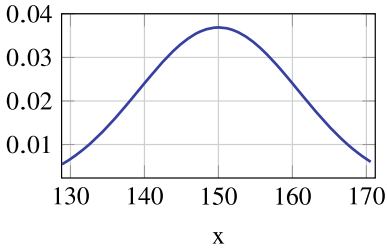


Fig. 2 Estimated density for the shortest path example

In the network of Fig. 1, there are six different directed paths from the source to the sink, each one being defined by a sequence of arcs. They are $\mathcal{P}_1 = \{1, 4, 10\}$, $\mathcal{P}_2 = \{1, 4, 8, 11\}, \mathcal{P}_3 = \{2, 5, 10\}, \mathcal{P}_4 = \{2, 5, 8, 11\}, \mathcal{P}_5 = \{2, 6, 9, 11\}$, and $\mathcal{P}_6 = \{3, 7, 9, 11\}$. The length of path p is $L_p = \sum_{j \in \mathcal{P}_p} Y_j$ and the length of the shortest path is

$$X = h(\boldsymbol{Y}) = \min_{1 \le p \le 6} L_p = \min_{1 \le p \le 6} \sum_{j \in \mathcal{P}_p} F_j^{-1}(U_j). \tag{9}$$

For our experiments, we assume that Y_j is normal with mean $\mu_j = 10j$ and standard deviation $\sigma_j = j$ (to make things simple). The probability of negative arc lengths is negligible. (To be mathematically cleaner, we can truncate the normal density to $[0, \infty)$, but it makes no visible difference in the numerical results.) We estimate the density of X over $[a, b] = [128.8, 171.2]$, which covers about 95% of the density. This density is shown in Fig. 2. It is close to a normal distribution, which is not surprising because all the Y_j are normal.

For the CDE, we select a directed minimal cut $\mathcal{L}$ between the source and the sink, and we condition on $\mathcal{G} = \{Y_j,\ j \notin \mathcal{L}\}$, similarly as for the SAN example in [17]. If $P_j + Y_j$ is the length of the shortest path that goes through arc j for $j \in \mathcal{L}$, then conditional on $\mathcal{G}$, each P_j is known and the conditional cdf of X is

$$F(x \mid \mathcal{G}) = \mathbb{P}\left[X \le x \mid \{P_j : j \in \mathcal{L}\}\right] = 1 - \prod_{j \in \mathcal{L}} (1 - F_j(x - P_j)). \tag{10}$$

If the Y_j's for $j \in \mathcal{L}$ are continuous variables, then the conditional density

$$f(x \mid \mathcal{G}) = \frac{\mathrm{d}}{\mathrm{d}x} F(x \mid \mathcal{G}) = \sum_{j \in \mathcal{L}} f_j(x - P_j) \prod_{l \in \mathcal{L}, l \ne j} (1 - F_l(x - P_l))$$

is an unbiased density estimator. In our numerical experiments, we try the two cuts $\mathcal{L}_1$ and $\mathcal{L}_2$ shown on the lower panel of Fig. 1.

For the LRDE we notice that $h(\boldsymbol{Y})$ is the minimum over the lengths of six possible paths. These lengths, in turn, are simple sums of several of the Y_j, so we have $h(c\boldsymbol{Y}) = ch(\boldsymbol{Y})$ for any constant $c > 0$. Therefore, with the change of variables $\varphi_x(\boldsymbol{z}) = x\,\boldsymbol{z}$ one obtains that $h(\varphi_x(\boldsymbol{z}))/x = h(\varphi_x(\boldsymbol{z})/x) = h(\boldsymbol{z})$ is independent of x.

For $x > 0$ this leads to the LRDE

$$\hat{f}_{\text{lrde}}(x) = \mathbb{I}[h(\boldsymbol{Y}) \le x]\; x^{-1} \left(-\sum_{j=1}^{s} (Y_j - \mu_j) Y_j \sigma_j^{-2} + s \right). \tag{11}$$

For GLR, the estimator in Proposition 4 does not apply to this example, because for the function h given in (9), for any choice of j, the required derivatives do not always exist. For the GLR-U, we want to find a subset of indices and a function g that satisfy the required conditions. In particular, $\tilde{g}$ must be a one-to-one continuous map between the selected inputs U_j and a selected subset of the path lengths L_p, so that the latter subset is sufficient to determine X and the Jacobian $J_g(\cdot; x)$ of this mapping is invertible. Note that the six path lengths are not independent: we have $L_1 + L_4 = L_2 + L_3$. But after removing one of these four paths, there is no linear relationship between any of the five L_p's that remain. Then we must select five input variables U_j for which the mapping g between those selected U_j's and the five L_p's is one-to-one when the other U_j's are fixed. There are several possibilities for the selection of these five indexes j for the inputs, each one leading to a different estimator. We will try two of them in our experiments, namely $\mathcal{J}_1 = \{1, 2, 3, 6, 8\}$ and $\mathcal{J}_2 = \{5, 7, 8, 10, 11\}$. Assuming that we remove the path $\mathcal{P}_4$ and select $\mathcal{J}_1$, we obtain $g(\boldsymbol{U}) = (g_1(\boldsymbol{U}), \ldots, g_5(\boldsymbol{U}))^t$ where $g_p(\boldsymbol{U}) = L_p$ for $p = 1, 2, 3$ and $g_p(\boldsymbol{U}) = L_{p+1}$ for $p = 4, 5$, and the Jacobian is computed by interpreting the $g_p(\boldsymbol{U})$ as functions of U_1, U_2, U_3, U_6, U_8 alone, with the other U_j's fixed. The GLR-U density estimator in (8) turns out to be

$$G_1(\boldsymbol{U}; x) = -\mathbb{I}[h(\boldsymbol{Y}) \le x] \sum_{j=1}^{3} j^{-1} \Phi^{-1}(U_j) = -\mathbb{I}[h(\boldsymbol{Y}) \le x] \sum_{j=1}^{3} j^{-2} (Y_j - 10j) \tag{12}$$

for $\mathcal{J}_1$ and

$$G_1(\boldsymbol{U}; x) = -\mathbb{I}[h(\boldsymbol{Y}) \le x] \sum_{j=10}^{11} j^{-1} \Phi^{-1}(U_j) = -\mathbb{I}[h(\boldsymbol{Y}) \le x] \sum_{j=10}^{11} j^{-2} (Y_j - 10j) \tag{13}$$

for $\mathcal{J}_2$, where Φ denotes the standard normal cdf.

Table 2 summarizes our numerical results for this example, for all the methods. It reports $-\log_2(\text{MISE})$ for $n = 2^{20}$ as well as the convergence rate exponent β for MC and its (noisy) estimate $\hat{\beta}$ for RQMC.

We find that the CDE combined with RQMC outperforms all other methods by a wide margin. Compared with the KDE with MC (the traditional approach), it reduces the MISE for $n = 2^{20}$ by a factor of about $2^{25} \approx 32$ millions. The orange cut $\mathcal{L}_2$ does better than the blue cut $\mathcal{L}_1$, especially for plain MC. This could appear surprising, because $\mathcal{L}_2$ has fewer arcs, but the explanation is that the two arcs of $\mathcal{L}_2$ have a much larger variance, so it pays off to hide them. Generally speaking, we want to select a

Table 2 Estimated values of $-\log_2(\text{MISE})$ with $n = 2^{20}$ points and estimated MISE rate $\hat{\beta}$ for various methods, for the shortest path example. The 21.3 entry (for example) means that for the KDE with MC and $n = 2^{20}$ points, we have MISE $\approx 2^{-21.3}$

Method	MC		RQMC	
	$-\log_2(\text{MISE})$	β	$-\log_2(\text{MISE})$	$\hat{\beta}$
KDE	21.3	0.8	25.7	0.96
CDE (blue cut)	24.7	1.0	45.6	2.12
CDE (orange cut)	29.1	1.0	46.5	1.66
LRDE	20.2	1.0	27.8	1.38
GLR-U in (12)	15.4	1.0	23.2	1.29
GLR-U in (13)	21.5	1.0	29.6	1.35

conditioning that hides (or integrate out) variables that capture as much variance as possible. (For the blue cut, the noise in the linear regression model and the estimate $\hat{\beta}$ appears quite significant.)

We also observe a significant difference of performance between the two choices of input variables for GLR-U. With $\mathcal{I}_2$, the performance is better than for the KDE, whereas for $\mathcal{I}_1$ it is worse. This shows that the choice of input variables may have a significant impact on the performance in general. Note that $\mathcal{I}_2$ contains input variables that have much more variance than $\mathcal{I}_1$. By comparing (12) and (13), we can see why the second estimator has less variance: the terms in the sum that multiplies the indicator have larger constants in the denominator, and therefore a smaller variance. In some sense, the GLR-U estimator integrates out part of the variance contained in the selected input variables, so it makes sense to select a subset of input variables that captures more of the variance.

The LRDE has a larger MISE than the KDE with $n = 2^{20}$ MC samples, but it beats the KDE when using RQMC. It also performs better than GLR-U for one choice of inputs and worse for the other choice.

With the same network, we now consider a slightly different problem. We assume that the Y_j's are random link capacities instead of random lengths, and we want to estimate the density of the maximum flow that can be sent from the source to the sink. This maximum flow $h(\boldsymbol{Y})$ is equal to the capacity of the minimal directed cut having the smallest capacity. Here, we assume that Y_j is normal with mean $\mu_j = 10$ and standard deviation $\sigma_j = 1$ for $j < 10$ and normal with mean $\mu_j = 20$ and standard deviation $\sigma_j = 4$ for $j = 10$ and 11. For the CDE, if we take $\mathcal{G}$ as in the previous case, the distribution of X conditional on $\mathcal{G}$ typically has a probability mass at some point. For instance, if $\mathcal{L} = \mathcal{L}_1$, then after the conditioning, $Y_{10} + Y_{11}$ is known and there is a positive probability that this is the value of the maximum flow. As a result, the conditional cdf is sometimes discontinuous and the CDE is no longer an unbiased density estimator. This motivates the use of LRDE for this example.

Similarly as in the previous example, $h(\boldsymbol{Y})$ is the minimum over several simple sums of Y_j's, so multiplying all Y_j's by a positive constant multiplies the maximum

Table 3 Values of the $\log_2(\text{MISE})$ estimated with $n = 2^{20}$ points and the estimated MISE rate $\hat{\beta}$ for various methods for the maximum flow example

Method	MC	RQMC	
	$-\log_2(\text{MISE})$	$-\log_2(\text{MISE})$	$\hat{\beta}$
KDE	18.3	19.7	0.86
LRDE	18.7	23.6	1.23
GLR-U	17.7	23.2	1.26

flow $h(\boldsymbol{Y})$ by the same constant. Therefore, the change of variables $\varphi_x(z) = xz$ can be used again and provides the exact same LRDE as in (11), but with the modified h, μ_j, and σ_j.

For GLR-U, the construction is similar as for the previous example, except that we select a subset of minimal cuts with independent capacities instead of a subset of paths. There are hundreds of thousands of ways of selecting the subset of minimal cuts. We tried a few of them and obtained the best results by selecting the set of cuts: {{10, 11}, {1, 2, 7}, {1, 2, 9}, {1, 2, 11}, {1, 5, 9}, {2, 3, 4}, {4, 5, 11}, {8, 9, 10}, {2, 3, 8, 10}, {6, 7, 8, 10}} and then hiding Y_{11}. This gives the estimator

$$G(\boldsymbol{U}; x) = -\mathbb{I}[h(\boldsymbol{Y}) \le x]\,((Y_1 - 10) + (Y_4 - 10) + (Y_{10} - 20)/16)\,.$$

Numerical results for the KDE, LRDE, and GLR-U for this example are given in Table 3. In terms of MISE, under MC, the LRDE performs better than GLR-U and slightly better than the KDE, but not much. However, RQMC improves the MISE for $n = 2^{20}$ by a factor of about 30 for the LRDE, a bit more for GLR-U, and about 3 for the KDE. The combination of LRDE or GLR-U with RQMC also improves the convergence rate $\hat{\beta}$.

9 Conclusion

We discussed and compared several recent developments regarding density estimation for simulation models, with Monte Carlo and quasi-Monte Carlo methods. Most of these methods provide unbiased density estimators and some of them are also RQMC-friendly, in which case their MISE can converge at a faster rate than the canonical rate of $O(1/n)$ as a function of the sample size n. For the classical density estimators in statistics, in contrast, the MISE converges at a slower rate than $O(1/n)$. In our numerical example (and several other experiments not reported here), the CDE combined with RQMC was by far the best performer. However, for some types of problems it may be difficult to apply, and then one can rely on one of the alternatives. In future work, these density estimators should be adapted, implemented, and compared for a larger variety of Monte Carlo applications for which density estimates are useful.

Acknowledgements This work has been supported by a NSERC Discovery Grant and an IVADO Grant to P. L'Ecuyer. F. Puchhammer was also supported by Spanish and Basque governments fundings through BCAM (ERDF, ESF, SEV-2017-0718, PID2019-108111RB-I00, PID2019-104927GB-C22, BERC 2018e2021, EXP. 2019/00432, ELKARTEK KK-2020/00049), and the computing infrastructure of i2BASQUE academic network and IZO-SGI SGIker (UPV).

References

1. Asmussen, S.: Conditional Monte Carlo for sums, with applications to insurance and finance. Ann. Actuar. Sci. **12**(2), 455–478 (2018)
2. Asmussen, S., Glynn, P.W.: Stochastic Simulation. Springer, New York (2007)
3. Ben Abdellah, A., L'Ecuyer, P., Owen, A., Puchhammer, F.: Density estimation by randomized Quasi-Monte Carlo. SIAM J. Uncertain. Quantif. **9**(1), 280–301 (2021)
4. Dick, J., Pillichshammer, F.: Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration. Cambridge University Press, Cambridge, U.K. (2010)
5. Fu, M., Hu, J.Q.: Conditional Monte Carlo: Gradient Estimation and Optimization Applications. Kluwer Academic, Boston (1997)
6. Fu, M.C.: Sensitivity analysis in Monte Carlo simulation of stochastic activity networks. In: Alt, F.B., Fu, M.C., Golden, B.L. (eds.) Perspectives in Operations Research, Operations Research/Computer Science Interfaces Series, pp. 351–366. Springer, Boston (2006)
7. Glynn, P.W.: Likelihood ratio gradient estimation: an overview. In: Proceedings of the 1987 Winter Simulation Conference, pp. 366–375. IEEE Press, Piscataway, NJ (1987)
8. Glynn, P.W., L'Ecuyer, P.: Likelihood ratio gradient estimation for regenerative stochastic recursions. Adv. Appl. Probab. **27**, 1019–1053 (1995)
9. Goda, T., Suzuki, K.: Recent Advances in Higher Order Quasi-Monte Carlo Methods, pp. 69–102. De Gruyter (2019)
10. Laub, P.J., Salomone, R., Botev, Z.I.: Monte Carlo estimation of the density of the sum of dependent random variables. Mathematics and Computers in Simulation **161**, 23–31 (2019)
11. L'Ecuyer, P.: A unified view of the IPA, SF, and LR gradient estimation techniques. Manag. Sci. **36**(11), 1364–1383 (1990)
12. L'Ecuyer, P.: Quasi-Monte Carlo methods with applications in finance. Financ. Stoch. **13**(3), 307–349 (2009)
13. L'Ecuyer, P.: Randomized Quasi-Monte Carlo: an introduction for practitioners. In: Glynn, P.W., Owen, A.B. (eds.) Monte Carlo and Quasi-Monte Carlo Methods: MCQMC 2016, pp. 29–52. Springer, Berlin (2018)
14. L'Ecuyer, P., Munger, D.: Algorithm 958: lattice builder: a general software tool for constructing rank-1 lattice rules. ACM Trans. Math. Softw. **42**(2), Article 15 (2016)
15. L'Ecuyer, P., Perron, G.: On the convergence rates of IPA and FDC derivative estimators. Oper. Res. **42**(4), 643–656 (1994)
16. Puchhammer, F., L'Ecuyer, P.: Likelihood Ratio Density Estimation for Simulation Models (2022). Submitted
17. L'Ecuyer, P., Puchhammer, F., Ben Abdellah, A.: Monte Carlo and Quasi-Monte Carlo density estimation via conditioning. INFORMS J. Comput. (2022). To appear. See https://doi.org/10.1287/ijoc.2021.1135
18. Lei, L., Peng, Y., Fu, M.C., Hu, J.Q.: Applications of generalized likelihood ratio method to distribution sensitivities and steady-state simulation. Discret. Event Dyn. Syst. **28**(1), 109–125 (2018)
19. Lemieux, C., Cieslak, M., Luttmer, K.: RandQMC User's Guide: A Package for Randomized Quasi-Monte Carlo Methods in C (2004). Software user's guide, available at http://www.math.uwaterloo.ca/~clemieux/randqmc.html

20. Niederreiter, H.: Random number generation and Quasi-Monte Carlo methods. In: SIAM CBMS-NSF CBMS Regional Conference Series in Mathematics, vol.63. SIAM (1992)
21. Parzen, E.: On estimation of a probability density function and mode. Ann. Math. Stat. **33**(3), 1065–1076 (1962)
22. Peng, Y., Fu, M.C., Heidergott, B., Lam, H.: Maximum likelihood estimation by Monte Carlo simulation: towards data-driven stochastic modeling. Oper. Res. **68**(6), 1896–1912 (2020)
23. Peng, Y., Fu, M.C., Hu, J.Q., Heidergott, B.: A new unbiased stochastic derivative estimator for discontinuous sample performances with structural parameters. Oper. Res. **66**(2), 487–499 (2018)
24. Peng, Y., Fu, M.C., Hu, J.Q., L'Ecuyer, P., Tuffin, B.: Generalized likelihood ratio method for stochastic models with uniform random numbers as inputs (2021). Submitted manuscript
25. Peng, Y., Fu, M.C., Hu, J.Q., L'Ecuyer, P., Tuffin, B.: Variance reduction for generalized likelihood ratio method by conditional Monte Carlo and randomized Quasi-Monte Carlo. J. Manage. Sci. Eng. (2022). To appear
26. Scott, D.W.: Multivariate Density Estimation, 2nd edn. Wiley, New York (2015)

Quasi-Monte Carlo Software

Sou-Cheng T. Choi, Fred J. Hickernell, Rathinavel Jagadeeswaran, Michael J. McCourt, and Aleksei G. Sorokin

Abstract Practitioners wishing to experience the efficiency gains from using low discrepancy sequences need correct, robust, well-written software. This article, based on our MCQMC 2020 tutorial, describes some of the better quasi-Monte Carlo (QMC) software available. We highlight the key software components required by QMC to approximate multivariate integrals or expectations of functions of vector random variables. We have combined these components in QMCPy, a Python open-source library, which we hope will draw the support of the QMC community. Here we introduce QMCPy.

Keywords Low discrepancy generators · Monte Carlo stopping criteria · Open-source software

S.-C. T. Choi
Kamakura Corporation, 2222 Kalakaua Ave, Suite 1400, Honolulu, HI 96815, USA
e-mail: schoi32@iit.edu

S.-C. T. Choi · F. J. Hickernell (✉) · R. Jagadeeswaran · A. G. Sorokin
Department of Applied Mathematics, Illinois Institute of Technology, RE 220, 10 W. 32nd St., Chicago, IL 60616, USA
e-mail: hickernell@iit.edu

R. Jagadeeswaran
e-mail: jrathin1@iit.edu

A. G. Sorokin
e-mail: asorokin@hawk.iit.edu

F. J. Hickernell
Center for Interdisciplinary Scientific Computation and Department of Applied Mathematics, Illinois Institute of Technology, RE 220, 10 W. 32nd St., Chicago, IL 60616, USA

R. Jagadeeswaran
Wi-Tronix LLC, 631 E Boughton Rd, Suite 240, Bolingbrook, IL 60440, USA

M. J. McCourt
SigOpt, an Intel company, 100 Bush St., Suite 1100, San Francisco, CA 94104, USA
e-mail: mccourt@sigopt.com

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_2

1 Introduction

Quasi-Monte Carlo (QMC) methods promise great efficiency gains over independent and identically distributed (IID) Monte Carlo (MC) methods. In some cases, QMC achieves one hundredth of the error of IID MC in the same amount of time (see Fig. 6). Often, these efficiency gains are obtained simply by replacing IID sampling with low discrepancy (LD) sampling, which is the heart of QMC.

Practitioners might wish to test whether QMC would speed up their computation. Access to the best QMC algorithms available would make that easier. Theoreticians or algorithm developers might want to demonstrate their ideas on various use cases to show their practical value.

This tutorial points to some of the best QMC software available. Then we describe QMCPy [6],[1] which is crafted to be a community-owned Python library that combines the best QMC algorithms and interesting use cases from various authors under a common user interface.

The model problem for QMC is approximating a multivariate integral,

$$\mu := \int_{\mathcal{T}} g(\boldsymbol{t})\, \lambda(\boldsymbol{t})\, \mathrm{d}\boldsymbol{t}, \tag{1}$$

where g is the integrand, and λ is a non-negative weight. If λ is a probability distribution (PDF) for the random variable $\boldsymbol{T}$, then μ is the mean of $g(\boldsymbol{T})$. Regardless, we perform a suitable variable transformation to interpret this integral as the mean of a function of a multivariate, standard uniform random variable:

$$\mu = \mathbb{E}[f(\boldsymbol{X})] = \int_{[0,1]^d} f(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, \qquad \boldsymbol{X} \sim \mathcal{U}[0,1]^d. \tag{2}$$

QMC approximates the population mean, μ, by a sample mean,

$$\widehat{\mu} := \frac{1}{n} \sum_{i=0}^{n-1} f(\boldsymbol{X}_i), \qquad \boldsymbol{X}_0, \boldsymbol{X}_1, \ldots \overset{\mathrm{M}}{\sim} \mathcal{U}[0,1]^d. \tag{3}$$

The choice of the sequence $\{\boldsymbol{X}_i\}_{i=0}^{\infty}$ and the choice of n to satisfy the prescribed error requirement,

$$|\mu - \widehat{\mu}| \le \varepsilon \qquad \text{absolutely or with high probability,} \tag{4}$$

are important decisions, which QMC software helps the user make.

[1] QMCPy is in active development. This article is based on version 1.2 on PyPI.

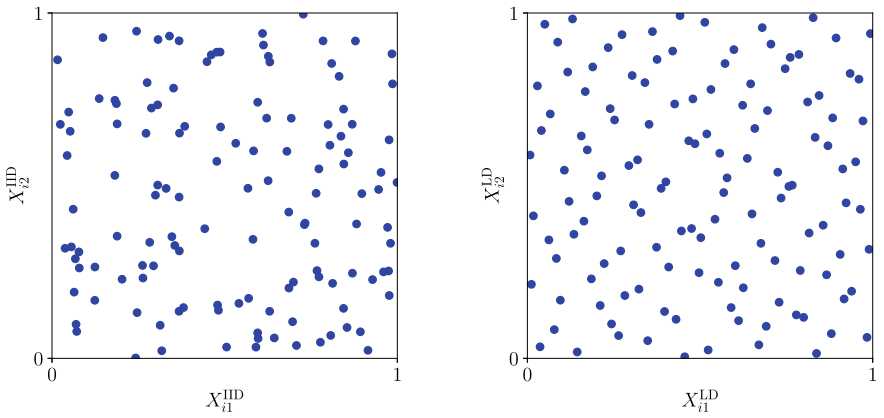


Fig. 1 IID points (left) contrasted with LD points (right). The LD points cover the square more evenly

Here, the notation $\overset{\text{M}}{\sim}$ means that the sequence mimics the specified, target distribution, but not necessarily in a probabilistic way. We use this notation in two forms: $\overset{\text{IID}}{\sim}$ and $\overset{\text{LD}}{\sim}$.

IID sequences must be random. The position of any point is not influenced by any other, so clusters and gaps occur. A randomly chosen subsequence of an IID sequence is also IID. When we say that $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots \overset{\text{IID}}{\sim} F$ for some distribution F, we mean that for any positive integer n, the multivariate probability distribution of $\boldsymbol{X}_0, \ldots, \boldsymbol{X}_{n-1}$ is the product of the marginals, specifically,

$$F_n(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) = F(\boldsymbol{x}_0) \cdots F(\boldsymbol{x}_{n-1}).$$

When IID points are used to approximate μ by the sample mean, the root mean squared error is $O(n^{-1/2})$. Figure 1 displays IID uniform points, $\boldsymbol{X}_0^{\text{IID}}, \boldsymbol{X}_1^{\text{IID}}, \ldots \overset{\text{IID}}{\sim} \mathcal{U}[0,1]^2$, i.e., the target distribution is $F_{\text{unif}}(\boldsymbol{x}) = x_1 x_2$.

LD sequences may be deterministic or random, but each point is carefully coordinated with the others so that they fill the domain well. Subsequences of LD sequences are generally not LD. When we say that $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots \overset{\text{LD}}{\sim} \mathcal{U}[0,1]^d$, we mean that for any positive integer n, the *empirical distribution* of $\boldsymbol{X}_0, \ldots, \boldsymbol{X}_{n-1}$, denoted $F_{\{\boldsymbol{X}_i\}_{i=0}^{n-1}}$, approximates the uniform distribution, F_{unif}, well (relative to n). (The empirical distribution of a set assigns equal probability to each point.).

A measure of the difference between the empirical distribution of a set of points and the uniform distribution is called a *discrepancy* and is denoted $D(\{\boldsymbol{X}_i\}_{i=0}^{n-1})$ [10, 17, 18, 38]. This is the origin of the term "low discrepancy" points or sequences. LD points by definition have a smaller discrepancy than IID points. Figure 1 contrasts IID uniform points with LD points, $\boldsymbol{X}_0^{\text{LD}}, \boldsymbol{X}_1^{\text{LD}} \ldots \overset{\text{LD}}{\sim} \mathcal{U}[0,1]^2$, in this case, linearly scrambled and digitally shifted Sobol' points.

The error in using the sample mean to approximate the integral can be bounded according to the Koksma-Hlawka inequality and its extensions [10, 17, 18, 38] as the product of the discrepancy of the sampling sequence and the variation of the integrand, denoted $V(\cdot)$:

$$|\mu - \widehat{\mu}| = \left| \int_{[0,1]^d} f(\boldsymbol{x}) \, \mathrm{d}\big(F_{\mathrm{unif}} - F_{\{\boldsymbol{X}_i\}_{i=0}^{n-1}}\big)(\boldsymbol{x}) \right| \leq D\big(\{\boldsymbol{X}_i\}_{i=0}^{n-1}\big) V(f). \tag{5}$$

The variation is a (semi-) norm of the integrand in a suitable Banach space. The discrepancy corresponds to the norm of the error functional for that Banach space. For typical Banach spaces, the discrepancy of LD points is $O(n^{-1+\epsilon})$, a higher convergence order than for IID points. For details, readers may refer to the references.

Here, we expect the reader to see in Fig. 1 that the LD points cover the integration domain more evenly than IID points. LD sampling can be thought of as a more even distribution of the sampling sites than IID. LD sampling is similar to stratified sampling. In the examples below, the reader will see the demonstrably smaller cubature errors arising from using LD points.

In the sections that follow, we first overview available QMC software. We next describe an architecture for good QMC software, i.e., what the key components are and how they should interact. We then describe how we have implemented this architecture in QMCPy. Finally, we summarize further directions that we hope QMCPy and other QMC software projects will take. Those interested in following or contributing to the development of QMCPy are urged to visit the GitHub repository at https://github.com/QMCSoftware/QMCSoftware.

We have endeavored to be as accurate as possible at the time of writing this article. We hope that progress in QMC software development will make this article happily obsolete in the coming years.

2 Available Software for QMC

QMC software spans LD sequence generators, cubatures, and applications. Here we review the better-known software, recognizing that some software overlaps multiple categories. Whenever applicable, we state each library's accessibility in QMCPy, or contrast its functionalities with QMCPy's—where we lag behind in QMCPy, we strive to catch up in the near future.

Software focusing on generating high-quality LD sequences and their generators includes the following, listed in alphabetical order:

BRODA Commerical and non-commercial software developed jointly with I. M. Sobol' in C++, MATLAB, and Excel [28]. BRODA can generate Sobol' sequences up to 65,536 dimensions. In comparison, QMCPy supports Sobol' sequences up to 21,201 dimensions.

Burkhardt Various QMC software for generating van der Corput, Faure, Halton, Hammersley, Niederreiter, or Sobol' sequences in C, C++, Fortran, MATLAB, or Python [4]. In QMCPy, we have implemented digital net, lattice, and Halton generators.

LatNet Builder The successor to *Lattice Builder* [34], this is a C++ library with Python and Java interfaces (in SSJ below) for generating vectors or matrices for lattices and digital nets [9, 33]. QMCPy contains a module for parsing the resultant vectors or matrices from LatNet Builder for compatibility with our LD point generators.

MATLAB Commercial software for scientific computing [51], which contains Sobol' and Halton sequences in the Statistics and Machine Learning Toolbox. Both generators can be applied jointly with the Parallel Computing Toolbox to accelerate their execution speed. The dimension of the Sobol' sequences is restricted to 1,111, which is relatively small, yet sufficient for most applications.

MPS Magic Point Shop contains lattices and Sobol' sequences in C++, Python, and MATLAB [39]. QMCPy started with MPS for developing LD generators.

Owen Owen's randomized Halton sequences with dimensions up to 1,000 [43] and scrambled Sobol' sequences with dimensions up to 21,021 [41, 44] in R. QMCPy supports Owen's Halton randomization method, and we plan to implement Owen's nested uniform scrambling for digital nets in the near future.

PyTorch Open-source Python library for deep learning, with unscrambled or scrambled Sobol' sequences [46, 47]. PyTorch enables seamless utilization of Graphics Processing Units (GPUs) or Field Programmable Gate Arrays (FPGAs).

QMC.jl LD Sequences in Julia [48]. Julia [3] is an interpreted language similar to Python and R in terms of ease of use, but is designed to run much faster.

qrng Randomized Sobol', Halton, and Korobov sequences in R [23]. The default Halton randomization in QMCPy utilizes the methods from qrng.

SciPy Scientific computing library in Python with Latin hypercube, Halton, and Sobol' generators [52].

TF Quant Finance Google's Tensorflow deep-learning library [1] specialized for financial modeling [12]. It contains lattice and Sobol' generators alongside with algorithms sped up with GPUs, FGPAs, or Tensor Processing Units (TPUs).

Software focusing on QMC cubatures and applications includes the following:

GAIL The Guaranteed Automatic Integration Library contains automatic (Q)MC stopping criteria in MATLAB [5, 14]. These are iterative procedures for one- or high-dimensional integration that take a user's input error tolerance(s) and determine the number of (Q)MC sampling points necessary to achieve user-desired accuracy (almost surely). Most of GAIL's (Q)MC functions, some with enhancements, are implemented in Python in QMCPy.

ML(Q)MC Multi-Level (Q)MC routines in C, C++, MATLAB, Python, and R [11]. We have ported ML(Q)MC functions to QMCPy.

MultilevelEstimators.jl ML(Q)MC methods in Julia [49]. The author, Pieterjan Robbe, has contributed cubature algorithms and use cases to QMCPy.

OpenTURNS Open source initiative for the Treatment of Uncertainties, Risks'N Statistics [40] written in C++ and Python, leveraging R statistical packages, as well as LAPACK and BLAS for numerical linear algebra.

QMC4PDE QMC for elliptic PDEs with random diffusion coefficients in Python [30].

SSJ Stochastic Simulation with the `hups` package in Java [32].

UQLab Framework for Uncertainty Quantification in MATLAB [36]. The core of UQLab is closed source, but a large portion of the library is open source. Recently, UQ[py]Lab, the beta release of UQLab with Python bindings, is available as Software as a Service (SaaS) via UQCloud [31].

The sections that follow describe QMCPy [6], which is our attempt to establish a framework for QMC software and to combine the best of the above software under a common user interface written in Python 3. The choice of language was determined by the desire to make QMC software accessible to a broad audience, especially the technology industry.

3 Components of QMC Software

QMC cubature can be summarized as follows. We want to approximate the expectation, μ, well by the sample mean, $\widehat{\mu}$, where (1), (2), and (3) combine to give

$$\mu := \int_{\mathcal{T}} g(\boldsymbol{t})\,\lambda(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} = \mathbb{E}[f(\boldsymbol{X})] = \int_{[0,1]^d} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \approx \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{X}_i) =: \widehat{\mu},$$
$$\boldsymbol{X} \sim \mathcal{U}[0,1]^d, \quad \boldsymbol{X}_0, \boldsymbol{X}_1, \ldots \overset{\mathrm{M}}{\sim} \mathcal{U}[0,1]^d. \tag{6}$$

Moreover, we want to satisfy the error requirement in (4). This requires four components, which we implement as QMCPy classes.

Discrete Distribution produces the sequence $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots$ that mimics $\mathcal{U}[0,1]^d$;

True Measure $\boldsymbol{t} \mapsto \lambda(\boldsymbol{t})\mathrm{d}\boldsymbol{t}$ defines the original measure, e.g., Gaussian or Lebesgue;

Integrand g defines the original integrand, and f defines the transformed version to fit the `DiscreteDistribution`; and

Stopping Criterion determines how large n should be to ensure that $|\mu - \widehat{\mu}| \le \varepsilon$ as in (4).

The software libraries referenced in Sect. 2 provide one or more of these components. QMCPy combines multiple examples of all these components under an object-oriented framework. Each example is implemented as a concrete class that realizes the properties and methods required by the abstract class for that component. The following sections detail descriptions and specific examples for each component.

Thorough documentation of all QMCPy classes is available in [7]. Demonstrations of how QMCPy works are given in Google Colab notebooks [15, 16]. The project may be installed from PyPI into a Python 3 environment via the command `pip install qmcpy`. In the code snippets that follow, we assume QMCPy has been imported alongside NumPy [13] via the following commands in a Python Console:

```
>>> import qmcpy as qp
>>> import numpy as np
```

4 Discrete Distributions

LD sequences typically mimic $\mathcal{U}[0,1]^d$. Good sequences mimicking other distributions are obtained by transformations as described in the next section. We denote by `DiscreteDistribution` the abstract class containing LD sequence generators. In most cases that have been implemented, the points $\boldsymbol{X}_0, \dots, \boldsymbol{X}_{n-1}$ have an empirical (discrete) distribution that closely approximates the uniform distribution, say, in the sense of discrepancy. We also envision future possible `DiscreteDistribution` objects that assign unequal weights to the sampling points. In any case, the term "discrete" refers to the fact that these are sequences of points (and weights), not continuous distributions.

QMCPy implements *extensible* LD sequences, i.e., those that allow practitioners to obtain and use $\boldsymbol{X}_n, \boldsymbol{X}_{n+1}, \dots$ without discarding $\boldsymbol{X}_0, \dots, \boldsymbol{X}_{n-1}$. Halton sequences do not have preferred sample sizes n, but extensible integration lattices and digital sequences in base b prefer n to be a power of b. For integration lattices and digital sequences, we have focused on base $b = 2$ since this is a popular choice and for convenience in generating extensible sequences.

Integration lattices and digital sequences in base 2 have an elegant group structure, which we summarize in Table 1. The addition operator[2] is $\oplus$, and its inverse is $\ominus$. The unshifted sequence is $\boldsymbol{Z}_0, \boldsymbol{Z}_1, \dots$ and the randomly shifted sequence is $\boldsymbol{X}_0, \boldsymbol{X}_1, \dots$

We illustrate lattice and Sobol' sequences using QMCPy. First, we create an instance of a d=2 dimensional `Lattice` object of the `DiscreteDistribution` abstract class. Then we generate the first eight (non-randomized) points in this lattice.

[2] The operator $\oplus$ is commonly used to denote exclusive-or, which is its meaning for digital sequences in base 2. However, we are using it here in a more general sense.

Table 1 Properties of lattices and digital net sequences. Note that they share group properties but also have distinctives

Define …	
$\boldsymbol{Z}_1, \boldsymbol{Z}_2, \boldsymbol{Z}_4, \ldots \in [0,1)^d$ chosen well	
$\boldsymbol{Z}_i := i_0\boldsymbol{Z}_1 \oplus i_1\boldsymbol{Z}_2 \oplus i_2\boldsymbol{Z}_4 \oplus i_3\boldsymbol{Z}_8 \oplus \cdots$ for $i = i_0 + i_1 2 + i_2 4 + i_3 8 + \cdots$, $i_\ell \in \{0,1\}$	
$\boldsymbol{X}_i := \boldsymbol{Z}_i \oplus \boldsymbol{\Delta}$, where $\boldsymbol{\Delta} \overset{\text{IID}}{\sim} [0,1)^d$	
Rank-1 Integration Lattices	Digital Nets
$\boldsymbol{t} \oplus \boldsymbol{x} := (\boldsymbol{t} + \boldsymbol{x}) \bmod \boldsymbol{1}$	$\boldsymbol{t} \oplus \boldsymbol{x} :=$ binary digitwise addition, $\oplus_{\text{dig}}$
require $\boldsymbol{Z}_{2^m} \oplus \boldsymbol{Z}_{2^m} = \boldsymbol{Z}_{\lfloor 2^{m-1} \rfloor}$ $\forall m \in \mathbb{N}_0$	
Then it follows that …	
$\mathcal{P}_m := \{\boldsymbol{Z}_0, \ldots, \boldsymbol{Z}_{2^m-1}\}$, $\boldsymbol{Z}_i \oplus \boldsymbol{Z}_j \in \mathcal{P}_m$	$\forall i, j, k \in \{0, \ldots, 2^m - 1\}$
$\mathcal{P}_{\boldsymbol{\Delta},m} := \{\boldsymbol{X}_0, \ldots, \boldsymbol{X}_{2^m-1}\}$, $\boldsymbol{X}_i \oplus \boldsymbol{X}_j \ominus \boldsymbol{X}_k \in \mathcal{P}_{\boldsymbol{\Delta},m}$	$\forall m \in \mathbb{N}_0$

```
>>> lattice = qp.Lattice(dimension=2, randomize=False)
>>> lattice.gen_samples(n=8)
"ParameterWarning:
    Non-randomized lattice sequence includes the origin"
array([[0.   , 0.   ],
       [0.5  , 0.5  ],
       [0.25 , 0.75 ],
       [0.75 , 0.25 ],
       [0.125, 0.375],
       [0.625, 0.875],
       [0.375, 0.125],
       [0.875, 0.625]])
```

The first three generators for this lattice are $\boldsymbol{Z}_1 = (0.5, 0.5)$, $\boldsymbol{Z}_2 = (0.25, 0.75)$, and $\boldsymbol{Z}_4 = (0.125, 0.375)$. One can check that $(\boldsymbol{Z}_2 + \boldsymbol{Z}_4) \bmod \boldsymbol{1} = (0.375, 0.125) = \boldsymbol{Z}_6$, as Table 1 specifies.

The random shift has been turned off above to illuminate the group structure. We normally include the randomization to ensure that there are no points on the boundary of $[0,1]^d$. Then, when points are transformed to mimic distributions such as the Gaussian, no LD points will be transformed to infinity. Turning off the randomization generates a warning when the `gen_samples` method is called.

Now, we generate Sobol' points using a similar process as we did for lattice points. Sobol' sequences are one of the most popular example of digital sequences.

```
>>> sobol = qp.Sobol(2, randomize=False)
>>> sobol.gen_samples(8, warn=False)
array([[0.   , 0.   ],
       [0.5  , 0.5  ],
       [0.25 , 0.75 ],
       [0.75 , 0.25 ],
       [0.125, 0.625],
       [0.625, 0.125],
       [0.375, 0.375],
       [0.875, 0.875]])
```

Here, $\boldsymbol{Z}_4$ differs from that for lattices, but more importantly, addition for digital sequences differs from that for lattices. Using digitwise addition for digital sequences, we can confirm that according to Table 1,

$$\begin{aligned}\boldsymbol{Z}_2 \oplus_{\text{dig}} \boldsymbol{Z}_4 &= (0.25, 0.75) \oplus_{\text{dig}} (0.125, 0.625)\\ &= ({}_2 0.010, {}_2 0.110) \oplus_{\text{dig}} ({}_2 0.001, {}_2 0.101) = ({}_2 0.011, {}_2 0.011)\\ &= (0.375, 0.375) = \boldsymbol{Z}_6.\end{aligned}$$

By contrast, if we construct a digital sequence using the generators for the lattice above with $\boldsymbol{Z}_2 = (0.25, 0.75)$, and $\boldsymbol{Z}_4 = (0.125, 0.375)$, we would obtain

$$\begin{aligned}\boldsymbol{Z}_6 = \boldsymbol{Z}_2 \oplus_{\text{dig}} \boldsymbol{Z}_4 &= ({}_2 0.010, {}_2 0.110) \oplus_{\text{dig}} ({}_2 0.001, {}_2 0.011)\\ &= ({}_2 0.011, {}_2 0.101) = (0.375, 0.625),\end{aligned}$$

which differs from the $\boldsymbol{Z}_6 = (0.375, 0.125)$ constructed for lattices. To emphasize, lattices and digital sequences are different, even if they share the same generators, $\boldsymbol{Z}_1, \boldsymbol{Z}_2, \boldsymbol{Z}_4, \ldots$.

The examples of `qp.Lattice` and `qp.Sobol` illustrate how QMCPy LD generators share a common user interface. The dimension is specified when the instance is constructed, and the number of points is specified when the `gen_samples` method is called. Following Python practice, parameters can be input without specifying their names if input in the prescribed order. QMCPy also includes Halton sequences and IID sequences, again deferring details to the QMCPy documentation [7].

A crucial difference between IID generators and LD generators is reflected in the behavior of generating n points. For an IID generator, asking for n points repeatedly gives different points each time because they are meant to be random and independent.

```
>>> iid = qp.IIDStdUniform(2); iid.gen_samples(1)
array([[0.40538109, 0.12255759]])
>>> iid.gen_samples(1)
array([[0.50913741, 0.11312201]])
```

Your output may look different depending on the seed used to generate these random numbers.

On the other hand, for an LD generator, asking for n points repeatedly gives *the same* points each time because they are meant to be the first n points of a specific LD sequence.

```
>>> lattice = qp.Lattice(2); lattice.gen_samples(1)
array([[0.2827584 , 0.36731649]])
>>> lattice.gen_samples(1)
array([[0.2827584 , 0.36731649]])
```

Here we allow the randomization so that the first point in the sequence is not the origin. To obtain the *next* n points, one may specify the start and ending indices of the sequence.

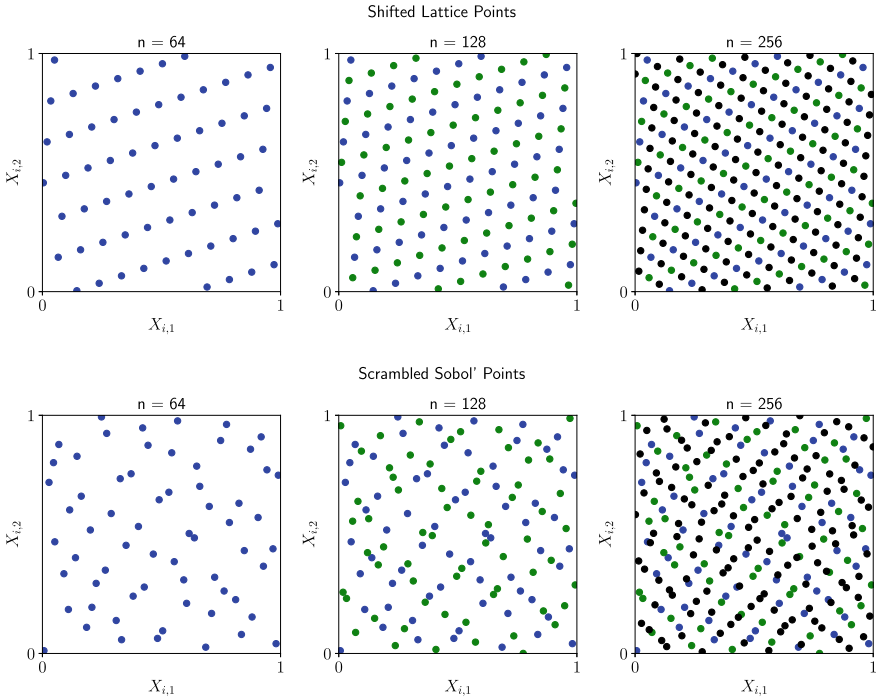


Fig. 2 Randomized lattice and Sobol' points mimicking a $\mathcal{U}[0,1]^2$ measure for $n = 64$, 128, and 256. Note how increasing the number of points evenly fills in the gaps between the points

```
>>> lattice.gen_samples(2)
array([[0.2827584 , 0.36731649],
       [0.7827584 , 0.86731649]])
>>> lattice.gen_samples(n_min=1, n_max=2)
array([[0.7827584 , 0.86731649]])
```

Figure 2 shows how increasing the number of lattice and Sobol' LD points through powers of two fills in the gaps in an even way.

5 True Measures

The LD sequences implemented as `DiscreteDistribution` objects mimic the $\mathcal{U}[0,1]^d$ distribution. However, we may need sequences to mimic other distributions. This is implemented via variable transformations, $\boldsymbol{\Psi}$. In general, if $\boldsymbol{X} \overset{\mathrm{M}}{\sim} \mathcal{U}[0,1]^d$, then

$$\boldsymbol{T} = \boldsymbol{\Psi}(\boldsymbol{X}) := \boldsymbol{a} + (\boldsymbol{b} - \boldsymbol{a}) \odot \boldsymbol{X} \overset{\mathrm{M}}{\sim} \mathcal{U}[\boldsymbol{a}, \boldsymbol{b}], \tag{7a}$$

$$\boldsymbol{T} = \boldsymbol{\Psi}(\boldsymbol{X}) := \boldsymbol{a} + \mathsf{A}\boldsymbol{\Phi}^{-1}(\boldsymbol{X}) \overset{\mathrm{M}}{\sim} \mathcal{N}(\boldsymbol{a}, \Sigma), \tag{7b}$$

$$\text{where } \boldsymbol{\Phi}^{-1}(\boldsymbol{X}) := \begin{pmatrix} \Phi^{-1}(X_1) \\ \vdots \\ \Phi^{-1}(X_d) \end{pmatrix}, \qquad \Sigma = \mathsf{A}\mathsf{A}^T,$$

and $\odot$ denotes term-by-term (Hadamard) multiplication. Here, $\boldsymbol{a}$ and $\boldsymbol{b}$ are assumed to be finite, and Φ is the standard Gaussian distribution function. Again we use $\overset{\mathrm{M}}{\sim}$ to denote mimicry, not necessarily in a probabilistic sense.

Figure 3 displays LD sequences transformed as described above to mimic a uniform and a Gaussian distribution. The code to generate these points takes the following form for uniform points based on a Halton sequence:

```
>>> u = qp.Uniform(
...          sampler = qp.Halton(2),
...          lower_bound = [-2, 0],
...          upper_bound = [2, 4])
>>> u.gen_samples(4)
array([[ 1.80379772,  3.51293599],
       [-0.19620228,  0.84626932],
       [ 0.80379772,  2.17960265],
       [-1.19620228,  3.95738043]])
```

whereas for Gaussian points based on a lattice sequence, we have:

```
>>> g = qp.Gaussian(qp.Lattice(2),
...           mean =          [3, 2],
...           covariance = [[9, 5],
...                         [5, 4]])
>>> g.gen_samples(4)
array([[-0.00920667,  0.29392389],
       [ 5.02422481,  1.45408341],
       [ 6.53688109,  5.14785917],
       [ 2.58403124,  1.16941969]])
```

Here the covariance decomposition $\Sigma = \mathsf{A}\mathsf{A}^T$ is done using principal component analysis. The Cholesky decomposition is also available.

The Brownian motion distribution arises often in financial risk applications. Here the d components of the variable $\boldsymbol{T}$ correspond to the discretized Brownian motion at times $\tau/d, 2\tau/d, \ldots, \tau$, where τ is the time horizon. The distribution is a special case of the Gaussian with covariance

$$\Sigma = (\tau/d)\big(\min(j,k)\big)_{j,k=1}^{d} \tag{8}$$

and mean $\boldsymbol{a}$, which is proportional to the times $(\tau/d)(1, 2, \ldots, d)^T$. The code for generating a Brownian motion is

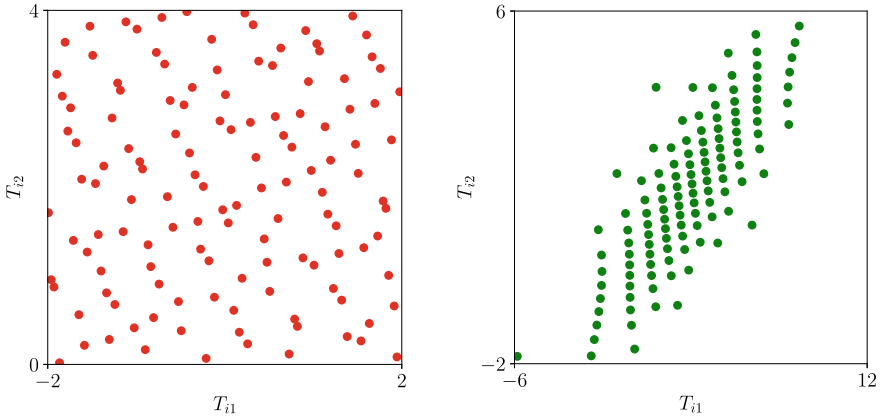


Fig. 3 Halton samples transformed to mimic a uniform $\mathcal{U}\left(\begin{bmatrix}-2\\0\end{bmatrix}, \begin{bmatrix}2\\4\end{bmatrix}\right)$ distribution (left) and lattice samples transformed to mimic a Gaussian $\mathcal{N}\left(\begin{bmatrix}3\\2\end{bmatrix}, \begin{bmatrix}9 & 5\\5 & 4\end{bmatrix}\right)$ distribution (right)

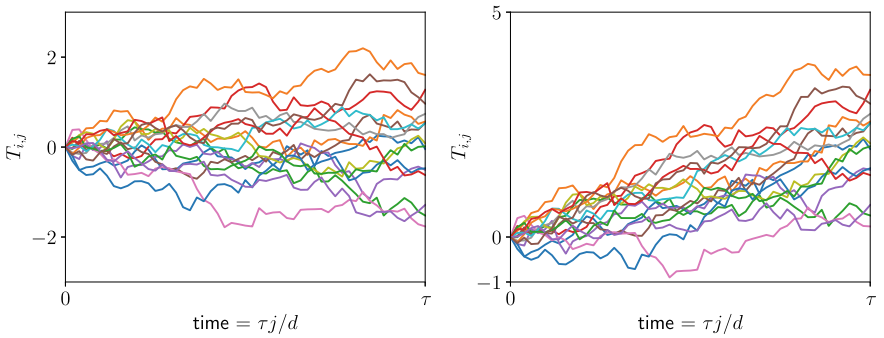


Fig. 4 Sobol' samples transformed to mimic a 52-dimensional Brownian Motion without drift (left) and with drift coefficient 2 (right)

```
>>> bm = qp.BrownianMotion(qp.Sobol(4), drift=2)
>>> bm.gen_samples(2)
array([[-0.28902829,  0.48582198,  1.81113976,  2.2376372 ],
       [ 1.41552651,  1.93926124,  1.27619672,  1.47496128]])
```

Figure 4 displays a Brownian motion based on Sobol' sequence with and without a drift.

6 Integrands

Let's return to the integration problem in (1), which we must rewrite as (2). We choose a transformation of variables defined as $\boldsymbol{t} = \boldsymbol{\Psi}(\boldsymbol{x})$ where $\boldsymbol{\Psi} : [0, 1]^d \to \mathcal{T}$. This leads to

$$\mu = \int_{\mathcal{T}} g(\boldsymbol{t})\,\lambda(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} = \int_{[0,1]^d} g\big(\boldsymbol{\Psi}(\boldsymbol{x})\big)\,\lambda\big(\boldsymbol{\Psi}(\boldsymbol{x})\big)\,|\boldsymbol{\Psi}'(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x} = \int_{[0,1]^d} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x},$$
$$\text{where } f(\boldsymbol{x}) = g\big(\boldsymbol{\Psi}(\boldsymbol{x})\big)\,\lambda\big(\boldsymbol{\Psi}(\boldsymbol{x})\big)\,|\boldsymbol{\Psi}'(\boldsymbol{x})|, \tag{9}$$

and $|\boldsymbol{\Psi}'(\boldsymbol{x})| := |\partial\boldsymbol{\Psi}/\partial\boldsymbol{x}|$ represents the Jacobian of the variable transformation. The abstract class `Integrand` provides f based on the user's input of g and the `TrueMeasure` instance, which defines λ and the transformation $\boldsymbol{\Psi}$. Different choices of $\boldsymbol{\Psi}$ lead to different f, which may give different rates of convergence of the cubature, $\widehat{\mu}$ to μ.

We illustrate the `Integrand` class via an example of Keister [27]:

$$\mu = \int_{\mathbb{R}^d} \cos(\|\boldsymbol{t}\|)\exp(-\boldsymbol{t}^T\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} = \int_{\mathbb{R}^d} \underbrace{\pi^{d/2}\cos(\|\boldsymbol{t}\|)}_{g(\boldsymbol{t})}\,\underbrace{\pi^{-d/2}\exp(-\boldsymbol{t}^T\boldsymbol{t})}_{\lambda(\boldsymbol{t})}\,\mathrm{d}\boldsymbol{t}. \tag{10}$$

Since λ is the density for $\mathcal{N}(\mathbf{0}, \mathsf{I}/2)$, it is natural to choose $\boldsymbol{\Psi}$ according to (7b) with $\mathsf{A} = \sqrt{1/2}\,\mathsf{I}$, in which case $\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})| = 1$, and so

$$\mu = \int_{[0,1]^d} \underbrace{\pi^{d/2}\cos(\|\boldsymbol{\Psi}(\boldsymbol{x})\|)}_{f(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x}, \qquad \boldsymbol{\Psi}(\boldsymbol{x}) := \sqrt{1/2}\,\boldsymbol{\Phi}^{-1}(\boldsymbol{x}).$$

The code below sets up an `Integrand` instance using QMCPy's `CustomFun` wrapper to tie a user-defined function g into the QMCPy framework. Then we evaluate the sample mean of $n = 1000$ f values obtained by sampling at transformed Halton points. Notice how a two-dimensional Halton generator is used to construct a Gaussian true measure, which is applied alongside the `my_Keister` function to instantiate a customized, QMCPy-compatible integrand for this problem.

```
>>> def my_Keister(t):
...     d = t.shape[1] # t is an (n x d) array
...     norm = np.sqrt((t**2).sum(1))
...     out = np.pi**(d/2)*np.cos(norm)
...     return out  # size n vector
...
>>> gauss = qp.Gaussian(qp.Halton(2), covariance=1/2)
>>> keister = qp.CustomFun(true_measure=gauss, g=my_Keister)
>>> x = keister.discrete_distrib.gen_samples(1000)
>>> y = keister.f(x)
>>> y.mean()
1.809055768468628
```

We have no indication yet of how accurate our approximation is. That topic is treated in the next section. Figure 5 visualizes sampling on the original integrand, g, and sampling on the transformed integrand, f.

Another way to approximate the Keister integral in (10) is to write it as an integral with respect to the Lebesgue measure:

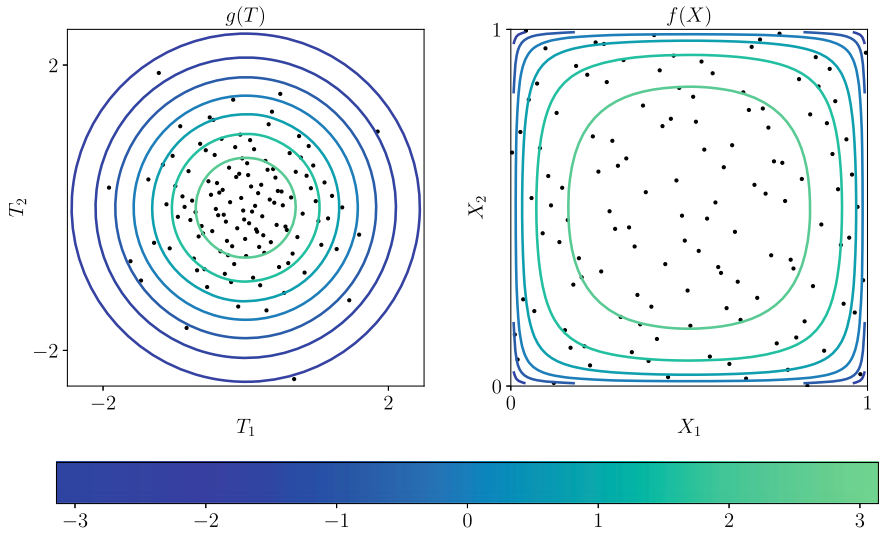


Fig. 5 Right: Sampling the transformed Keister integrand f at Halton points $\boldsymbol{X}_i \overset{\text{LD}}{\sim} \mathcal{U}[0,1]^2$. Left: Sampling the original Keister integrand g at $\boldsymbol{T}_i = \boldsymbol{\Psi}(\boldsymbol{X}_i) \overset{\text{M}}{\sim} \mathcal{N}(\boldsymbol{0}, \mathsf{I}/2)$ where $\boldsymbol{\Psi}$ is defined in (7b)

$$\begin{aligned}\mu &= \int_{\mathbb{R}^d} \underbrace{\cos(\|\boldsymbol{t}\|)\exp(-\boldsymbol{t}^T\boldsymbol{t})}_{g(\boldsymbol{t})} \underbrace{1}_{\lambda(\boldsymbol{t})}\,\mathrm{d}\boldsymbol{t} \\ &= \int_{[0,1]^d} \underbrace{\cos(\|\boldsymbol{\Psi}(\boldsymbol{x})\|)\exp(-\boldsymbol{\Psi}^T(\boldsymbol{x})\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})|}_{f(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x},\end{aligned}$$

where $\boldsymbol{\Psi}$ is any transformation from $[0,1]^d$ to $\mathbb{R}^d$. Now λ is *not* a PDF. QMCPy can perform the cubature this way as well.

```
>>> def my_L_Keister(t):
...     norm_sq = (t**2).sum(1)
...     out = np.cos(np.sqrt(norm_sq))*np.exp(-norm_sq)
...     return out
...
>>> lebesgue_gauss = qp.Lebesgue(qp.Gaussian(qp.Halton(2)))
>>> keister = qp.CustomFun(lebesgue_gauss, my_L_Keister)
>>> x = keister.discrete_distrib.gen_samples(1000)
>>> y = keister.f(x)
>>> y.mean()
1.8056340581961572
```

The $\boldsymbol{\Psi}$ chosen when transforming uniform sequences on the unit cube to fill $\mathbb{R}^d$ is given by (7b) with $\mathsf{A} = \mathsf{I}$.

In the examples above, one must input the correct g into `CustomFun` along with the correct `TrueMeasure` λ to define the integration problem. The `Keister` integrand included in the QMCPy library takes a more flexible approach to defining the integration problem μ in (10). Selecting a different `sampler` $\boldsymbol{\Psi}$ performs *importance sampling*, which leaves μ unchanged.

```
>>> # default transform
>>> keister = qp.Keister(qp.Halton(2))
>>> x = keister.discrete_distrib.gen_samples(1e4)
>>> keister.f(x).mean()
1.8082377673556123
>>> # custom transform for importance sampling
>>> keister = qp.Keister(sampler=qp.Gaussian(qp.Halton(2)))
>>> x = keister.discrete_distrib.gen_samples(1e4)
>>> keister.f(x).mean()
1.8080555069060817
```

In the first case above, the λ in (9) corresponds to the Gaussian density with mean zero and variance $1/2$ by default, and the corresponding variable transformation, $\boldsymbol{\Psi}$, is chosen to make $\lambda\big(\boldsymbol{\Psi}(\boldsymbol{x})\big)|\boldsymbol{\Psi}'(\boldsymbol{x})| = 1$ and $f(\boldsymbol{x}) = g\left(\boldsymbol{\Psi}(\boldsymbol{x})\right)$. In the second case, we choose an importance sampling density λ_{IS}, corresponding to standard Gaussian, and the variable transformation $\boldsymbol{\Psi}_{\mathrm{IS}}$ makes $\lambda_{\mathrm{IS}}\big(\boldsymbol{\Psi}_{\mathrm{IS}}(\boldsymbol{x})\big)|\boldsymbol{\Psi}'_{\mathrm{IS}}(\boldsymbol{x})| = 1$. Then

$$\begin{aligned}
\mu &= \int_{\mathcal{T}} g(\boldsymbol{t})\,\lambda(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} = \int_{\mathcal{T}} g(\boldsymbol{t})\,\frac{\lambda(\boldsymbol{t})}{\lambda_{\mathrm{IS}}(\boldsymbol{t})}\,\lambda_{\mathrm{IS}}(\boldsymbol{t})\mathrm{d}\boldsymbol{t} \\
&= \int_{[0,1]^d} g\big(\boldsymbol{\Psi}_{\mathrm{IS}}(\boldsymbol{x})\big)\,\frac{\lambda\big(\boldsymbol{\Psi}_{\mathrm{IS}}(\boldsymbol{x})\big)}{\lambda_{\mathrm{IS}}\big(\boldsymbol{\Psi}_{\mathrm{IS}}(\boldsymbol{x})\big)}\lambda_{\mathrm{IS}}\big(\boldsymbol{\Psi}_{\mathrm{IS}}(\boldsymbol{x})\big)\,|\boldsymbol{\Psi}'_{\mathrm{IS}}(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x} \\
&= \int_{[0,1]^d} f_{\mathrm{IS}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \\
&\qquad \text{where } f_{\mathrm{IS}}(\boldsymbol{x}) = g\big(\boldsymbol{\Psi}_{\mathrm{IS}}(\boldsymbol{x})\big)\,\frac{\lambda\big(\boldsymbol{\Psi}_{\mathrm{IS}}(\boldsymbol{x})\big)}{\lambda_{\mathrm{IS}}\big(\boldsymbol{\Psi}_{\mathrm{IS}}(\boldsymbol{x})\big)}. \qquad (11)
\end{aligned}$$

Because LD samples mimic $\mathcal{U}[0,1]^d$, choosing a different `sampler` is equivalent to choosing a different variable transform.

7 Stopping Criteria

The `StoppingCriterion` object determines the number of samples n that are required for the sample mean approximation $\widehat{\mu}$ to be within error tolerance ε of the true mean μ. Several QMC stopping criteria have been implemented in QMCPy, including replications, stopping criteria that track the decay of the Fourier complex exponential or Walsh coefficients of the integrand [20, 21, 26], and stopping criteria based on Bayesian credible intervals [24, 25].

The `CubQMCSobolG` stopping criterion used in the example below assumes the Walsh-Fourier coefficients of the integrand are absolutely convergent. The algorithm iteratively doubles the number of samples used in the integration and estimates the error using the decay of the Walsh coefficients [20]. When the estimated error is below the user-specified tolerance, it finishes the computation and returns the estimated integral.

Let us return to the Keister example from the previous section. After setting up a default `Keister` instance via a Sobol' `DiscreteDistribution`, we choose a `StoppingCriterion` object that matches the `DiscreteDistribution` and input our desired tolerance. Calling the `integrate` method returns the approximate integral plus some useful information about the computation.

```
>>> keister = qp.Keister(qp.Sobol(2))
>>> stopping = qp.CubQMCSobolG(keister, abs_tol=1e-3)
>>> solution_qmc,data_qmc = stopping.integrate()
>>> data_qmc # equivalent to print(data_qmc)
LDTransformData (AccumulateData Object)
    solution        1.808
    error_bound     6.06e-04
    n_total         2^(13)
    time_integrate  0.008
CubQMCSobolG (StoppingCriterion Object)
    abs_tol         0.001
    rel_tol         0
    n_init          2^(10)
    n_max           2^(35)
Keister (Integrand Object)
Gaussian (TrueMeasure Object)
    mean            0
    covariance      2^(-1)
    decomp_type     PCA
Sobol (DiscreteDistribution Object)
    d               2^(1)
    dvec            [0 1]
    randomize       LMS_DS
    graycode        0
    entropy         326942311248945520670220938885737472885
    spawn_key       ()
```

The second output of the stopping criterion provides helpful diagnostic information. This computation requires $n = 2^{13}$ Sobol' points and 0.008 seconds to complete. The error bound is 0.000606, which falls below the absolute tolerance.

QMC, which uses LD sequences, is touted as providing substantially greater computational efficiency compared to IID MC. Figure 6 compares the time and sample sizes needed to compute the 5-dimensional Keister integral (10) using IID sequences and LD lattice sequences. Consistent with what is stated in Section 1, the error of IID MC is $O(n^{-1/2})$, which means that the time and sample size to obtain an absolute error tolerance of ε is $O(\varepsilon^{-2})$. By contrast, the error of QMC using LD sequences is $O(n^{-1+\epsilon})$, which implies $O(\varepsilon^{-1-\epsilon})$ times and sample sizes. We see that QMC methods often require orders of magnitude fewer samples than MC methods to achieve the same error tolerance.

For another illustration of QMC cubature, we turn to pricing an Asian arithmetic mean call option. The (continuous-time) Asian option is defined in terms of the average of the stock price, which is written in terms of an integral. The payoff of this option is the positive difference between the strike price, K, averaged over the time horizon:

$$\text{payoff}(\boldsymbol{S}) = \max\left(\frac{1}{2d}\sum_{j=1}^{d}(S_{j-1}+S_j) - K,\ 0\right), \quad \boldsymbol{S} = (S_0, \ldots, S_d).$$

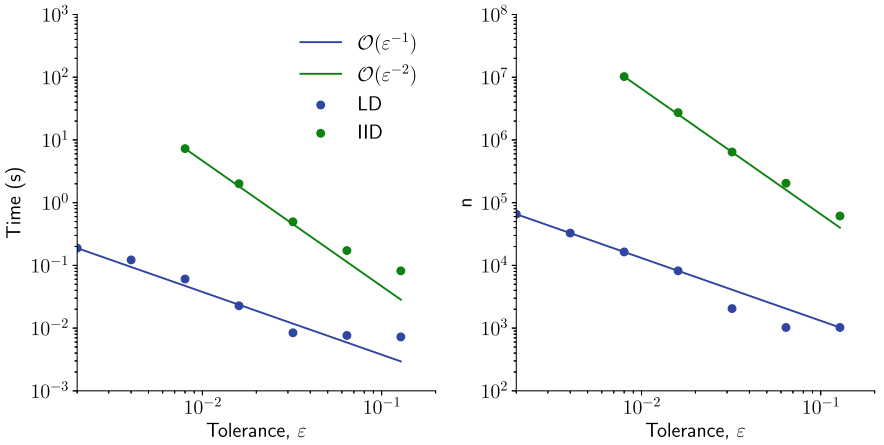


Fig. 6 Comparison of run times and sample sizes for computing the 5-dimensional Keister integral (10) using IID and LD lattice sequences for a variety of absolute error tolerances. The respective stopping criteria are `qp.CubMCG` [19] and `qp.CubQMCLatticeG` [26]. The LD sequences provide the desired answer much more efficiently

Here S_j denotes the asset price at time $\tau j/d$, and a trapezoidal rule is used for discrete approximation of the integral in time that defines the average. The trapezoidal rule is a more accurate approximation to the integral than a rectangle rule. A basic model for asset prices is a geometric Brownian motion,

$$S_j(\boldsymbol{T}) = S_0 \exp((r - \sigma^2/2)\tau j/d + \sigma T_j), \quad j = 1, \ldots, d, \quad \boldsymbol{T} = (T_1, \ldots, T_d) \sim \mathcal{N}(\boldsymbol{0}, \Sigma),$$

where Σ is defined in (8), r is the interest rate, σ is the volatility, and S_0 is the initial asset price. The fair price of the option is then the expected value of the discounted payoff, namely,

$$\text{price} = \mu = \mathbb{E}[g(\boldsymbol{T})], \quad \text{where } g(\boldsymbol{t}) = \text{payoff}\big(\boldsymbol{S}(\boldsymbol{t})\big) \exp(-r\tau).$$

The following code utilizes QMCPy's Asian option `Integrand` object to approximate the value of an Asian call option for a particular choice of the parameters.

```
>>> payoff = qp.AsianOption(
...     qp.Sobol(52), # weekly monitoring
...     start_price = 100,
...     strike_price = 120,
...     volatility = 0.5,
...     interest_rate = 0,
...     t_final = 1,
...     call_put = "call")
>>> qmc_stop_crit = qp.CubQMCSobolG(payoff, abs_tol=0.001)
>>> price,data = qmc_stop_crit.integrate()
>>> print("Option price = $%.3f using %.3f seconds, %.2e samples"
...     %(price, data.time_integrate, data.n_total))
Option price = $5.194 using 0.587 seconds, 1.31e+05 samples
```

Because this `Integrand` object has the built-in Brownian motion `TrueMeasure`, one only need provide the LD sampler.

Out of the money option price calculations can be sped up by adding an upward drift to the Brownian motion. The upward drift produces more in the money paths and also reduces the variation or variance of the final integrand, f. This is a form of importance sampling. Using a Brownian motion without drift we get

```
>>> payoff = qp.AsianOption(qp.Sobol(52),
...     start_price=100, strike_price=200)
>>> qmc_stopper = qp.CubQMCSobolG(payoff, abs_tol=0.001)
>>> price,data = qmc_stopper.integrate()
>>> print("Option price = $%.4f using %.3f seconds, n = %.2e"
...     %(price, data.time_integrate, data.n_total))
Option price = $0.1757 using 0.583 seconds, n = 1.31e+05
```

Adding the upward drift gives us the answer faster:

```
>>> payoff = qp.AsianOption(
...     sampler = qp.BrownianMotion(qp.Sobol(52), drift=1),
...     start_price=100, strike_price=200)
>>> qmc_stopper = qp.CubQMCSobolG(payoff, abs_tol=0.001)
>>> price,data_drift = qmc_stopper.integrate()
>>> print("Option price = $%.4f using %.3f seconds, n = %.2e"
...     %(price, data_drift.time_integrate, data_drift.n_total))
Option price = $0.1754 using 0.085 seconds, n = 1.64e+04
>>> print("Using drift required %.0f%% the time, %.0f%% the n"
...     %(100*data_drift.time_integrate / data.time_integrate,
...       100*data_drift.n_total / data.n_total))
Using drift required 15% the time, 12% the n
```

The choice of a good drift is an art.

The improvement in time is less than that in n because the integrand is more expensive to compute when the drift is employed. Referring to (11), in the case of no drift, the λ corresponds to the density for the discrete Brownian motion, and the variable transformation $\boldsymbol{\Psi}$ is chosen so that $f(\boldsymbol{x}) = g(\boldsymbol{\Psi}(\boldsymbol{x}))$. However, in the case of a drift, the integrand becomes $f_{\text{IS}}(\boldsymbol{x}) = g\big(\boldsymbol{\Psi}_{\text{IS}}(\boldsymbol{x})\big)\lambda\big(\boldsymbol{\Psi}_{\text{IS}}(\boldsymbol{x})\big)/\lambda_{\text{IS}}\big(\boldsymbol{\Psi}_{\text{IS}}(\boldsymbol{x})\big)$, which requires more computation time per integrand value.

8 Under the Hood

In this section, we look at the inner workings of QMCPy and point out features we hope will benefit the community of QMC researchers and practitioners. We also highlight important nuances of QMC methods and how QMCPy addresses these challenges. For details, readers should refer to the QMCPy documentation [7].

8.1 LD Sequences

LD sequences are the backbone of QMC methods. QMCPy provides generators that combine research from across the QMC community to enable advanced features and customization options.

Two popular LD sequences are integration lattices and digital nets which we previously outlined in Table 1. These LD generators are comprised of two parts: the static generating vectors $\boldsymbol{Z}_1, \boldsymbol{Z}_2, \boldsymbol{Z}_4, \ldots \in [0,1)^d$ and the callable generator function. By default, QMCPy provides a number of high-quality generating vectors for users to choose from. For instance, the default ordinary lattice vector was constructed by Cools, Kuo, and Nuyens [8] using component-by-component search, is extensible, has order-2 weights, and supports up to 3600 dimensions and 2^{20} samples. However, users who require more samples but fewer dimensions may switch to a generating vector constructed using LatNet Builder [9, 33] to support 750 dimensions and 2^{24} samples. Moreover, the `qp.Lattice` and `qp.DigitalNet` objects allow users to input their own generating vectors to produce highly customized sequences. To find such vectors, we recommend using LatNet Builder's construction routines as the results can be easily parsed into a QMCPy-compatible format.

Along with the selection of a generating vector, QMCPy's low discrepancy sequence routines expose a number of other customization parameters. For instance, the lattice generator extends the Magic Point Shop [39] to support either linear or natural ordering. Digital sequences permit either standard or Gray code ordering and may be randomized via a digital shift optionally combined with a linear scrambling [37]. Halton sequences may be randomized via the routines of either Owen [43] or Hofert and Lemieux [23].

8.2 A Word of Caution When Using LD Sequences

Although QMCPy's `DiscreteDistributions` have many of the same parameters and methods, users should be careful when swapping IID sequences with LD sequences. While IID node-sets have no preferred sample size, LD sequences often require special sampling ranges to ensure optimal discrepancy. As mentioned earlier, base-2 digital nets and extensible integration lattices show better evenness for sample sizes that are powers of 2. On the other hand, the preferred sample sizes for d-dimensional Halton sequences are $n = \prod_{j=1}^{d} p_j^{m_j}$ where p_j is the jth prime number and $m_j \in \mathbb{N}_0$ for $j = 1, \ldots, d$. Due to the infrequency of such values, Halton sequences are often regarded as not having a preferred sample size.

Users may also run into trouble when trying to generate too many points. Since QMCPy's generators construct sequences in 32-bit precision, generating greater than 2^{32} consecutive samples will cause the sequence to repeat. In the future, we plan to expand our generators to support optional 64-bit precision at the cost of greater computational overhead.

Another subtlety arises when transforming LD sequences to mimic different distributions. As mentioned earlier, unrandomized lattice and digital sequences include the origin, making transformations such as (7b) produce infinite values.

Some popular implementations of LD sequences drop the first point, which is the origin in the absence of randomization. The rationale is to avoid the transformation of the origin to infinity when mimicking a Gaussian or other distribution with an infinite

sample space. Unfortunately, dropping the first point destroys some nice properties of the first $n = 2^m$ points of LD sequences, which can degrade the order of convergence for QMC cubature. A careful discussion of this matter is given by [42].

8.3 Transformations

The transformation $\boldsymbol{\Psi}$ connects a `DiscreteDistribution` and `TrueMeasure`. So far, we have assumed the `DiscreteDistribution` mimics a $\mathcal{U}[0,1]^d$ distribution with PDF $\varrho(\boldsymbol{x}) = 1$. However, it may be advantageous to utilize a `DiscreteDistribution` that mimics a different distribution.

Suppose we have a `DiscreteDistribution` mimicking density ϱ supported on $\mathcal{X}$. Then using the variable transformation $\boldsymbol{\Psi}$,

$$\mu = \int_{\mathcal{T}} g(\boldsymbol{t})\,\lambda(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} = \int_{\mathcal{X}} f(\boldsymbol{x})\,\varrho(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \quad \text{for } f(\boldsymbol{x}) = g\big(\boldsymbol{\Psi}(\boldsymbol{x})\big)\,\frac{\lambda\big(\boldsymbol{\Psi}(\boldsymbol{x})\big)}{\varrho(\boldsymbol{x})}\,|\boldsymbol{\Psi}'(\boldsymbol{x})|,$$

which generalizes (9). QMCPy also includes support for successive changes of measures so users may build complex variable transformations in an intuitive manner. Suppose that the variable transformation is a composition of several transformations: $\boldsymbol{\Psi} = \widehat{\boldsymbol{\Psi}}_L = \boldsymbol{\Psi}_L \circ \boldsymbol{\Psi}_{L-1} \circ \cdots \circ \boldsymbol{\Psi}_1$ as in (11). Here, $\boldsymbol{\Psi}_l : \mathcal{X}_{l-1} \to \mathcal{X}_l$, $\mathcal{X}_0 = \mathcal{X}$, and $\mathcal{X}_L = \mathcal{T}$ so that the transformations are compatible with the `DiscreteDistribution` and `TrueMeasure`. Let $\widehat{\boldsymbol{\Psi}}_l = \boldsymbol{\Psi}_l \circ \boldsymbol{\Psi}_{l-1} \circ \cdots \circ \boldsymbol{\Psi}_1$ denote the composition of the first l transforms and assume that $\widehat{\boldsymbol{\Psi}}_0(\boldsymbol{x}) = \boldsymbol{x}$, the identity transform. Then we may write $\mu = \int_{\mathcal{X}} f(\boldsymbol{x})\,\varrho(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ for

$$f(\boldsymbol{x}) = g\big(\widehat{\boldsymbol{\Psi}}_L(\boldsymbol{x})\big)\frac{\lambda\big(\widehat{\boldsymbol{\Psi}}_L(\boldsymbol{x})\big)}{\varrho(\boldsymbol{x})}\prod_{l=1}^{L}\big|\boldsymbol{\Psi}_l'\big(\widehat{\boldsymbol{\Psi}}_{l-1}(\boldsymbol{x})\big)\big|.$$

It is often the case that $\boldsymbol{\Psi}_l$ is chosen such that $\boldsymbol{\Psi}_l(\boldsymbol{X})$ is stochastically equivalent to a random variable with density λ_l on sample space $\mathcal{X}_l$ when $\boldsymbol{X}$ is a random variable with density ϱ_l on sample space $\mathcal{X}_{l-1}$. This implies $\varrho_l(\boldsymbol{x}) = \lambda_l(\boldsymbol{\Psi}_l(\boldsymbol{x}))|\boldsymbol{\Psi}_l'(\boldsymbol{x})|$ so that

$$f(\boldsymbol{x}) = g\big(\widehat{\boldsymbol{\Psi}}_L(\boldsymbol{x})\big)\frac{\lambda\big(\widehat{\boldsymbol{\Psi}}_L(\boldsymbol{x})\big)}{\varrho(\boldsymbol{x})}\prod_{l=1}^{L}\frac{\varrho_l(\widehat{\boldsymbol{\Psi}}_{l-1}(\boldsymbol{x}))}{\lambda_l(\widehat{\boldsymbol{\Psi}}_l(\boldsymbol{x}))}.$$

For an example, we return to the Keister integral (10). The following code constructs three `Keister` instances: one without importance sampling, one importance sampled by a Gaussian distribution, and one importance sampled by the composition of a Gaussian distribution with a Kumaraswamy distribution [29]. All `Integrands` use a Sobol' `DiscreteDistribution`, making $\varrho(\boldsymbol{x}) = 1$ and $\mathcal{X} = [0,1]^d$. The `TrueMeasure` is $\mathcal{N}(\boldsymbol{0}, \mathsf{I}/2)$ making $\lambda(\boldsymbol{t}) = \pi^{-d/2}\exp(-\boldsymbol{t}^T\boldsymbol{t})$ and $\mathcal{T} = \mathbb{R}^d$.

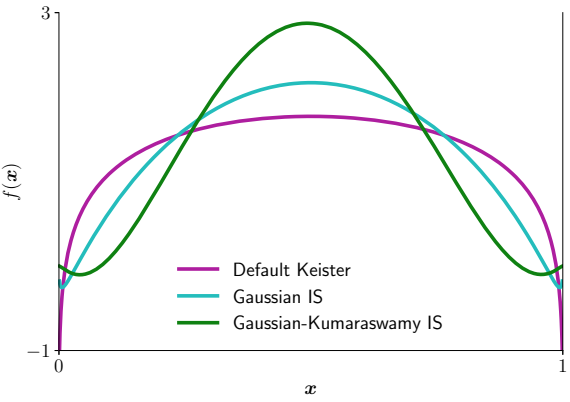


Fig. 7 Keister functions with and without importance sampling (`IS`). Note that the Keister functions using importance sampling are generally less variable and therefore easier to integrate, as evidenced by the faster integration times

The table below displays the variable transformations and the measures for these three cases. In all cases $\varrho_1(\boldsymbol{x}) = \cdots = \varrho_L(\boldsymbol{x}) = 1$ because the $\boldsymbol{\Psi}_l$ utilize inverse cumulative distributions.

Integrand	L	λ_1	$\boldsymbol{\Psi}_1$	λ_2	$\boldsymbol{\Psi}_2$	f
K	1	$\mathcal{N}(\mathbf{0}, \mathsf{I}/2)$	(7b)			$g(\boldsymbol{\Psi}_1(\cdot))$
K_gauss	1	$\mathcal{N}(\mathbf{0}, 3\mathsf{I}/4)$	(7b)			$g(\boldsymbol{\Psi}_1(\cdot))\dfrac{\lambda(\boldsymbol{\Psi}_1(\cdot))}{\lambda_1(\boldsymbol{\Psi}_1(\cdot))}$
K_gauss_kuma	2	Kum	$\boldsymbol{F}_{\text{Kum}}^{-1}$	$\mathcal{N}(\mathbf{0}, \mathsf{I})$	(7b)	$\dfrac{g(\boldsymbol{\Psi}_2(\boldsymbol{\Psi}_1(\cdot)))\lambda(\boldsymbol{\Psi}_2(\boldsymbol{\Psi}_1(\cdot)))}{\lambda_1(\boldsymbol{\Psi}_1(\cdot))\lambda_2(\boldsymbol{\Psi}_2(\boldsymbol{\Psi}_1(\cdot)))}$

Here Kum denotes the multivariate Kumaraswamy distribution with independent marginals, and $\boldsymbol{F}_{\text{Kum}}^{-1}$ denotes the element-wise inverse cumulative distribution function. The code below evaluates the Keister integral (10) for $d = 1$ and error tolerance $\varepsilon = 5 \times 10^{-8}$. The timings for each of these different integrands are displayed.

```
>>> sobol = qp.Sobol(1)
>>> K = qp.Keister(sobol) # keister 0
>>> K_gauss = qp.Keister( # keister 1
...     qp.Gaussian(sobol, covariance=.75))
>>> K_gauss_kuma = qp.Keister( # keister 2
...     qp.Gaussian(
...         qp.Kumaraswamy(sobol, a=.8, b=.8)))
>>> for i,keister in enumerate([K, K_gauss, K_gauss_kuma]):
...     stopper = qp.CubQMCSobolG(keister, abs_tol=5e-8)
...     sol,data = stopper.integrate()
...     print("keister %d integration time = %.3f seconds"
...         %(i, data.time_integrate))
keister 0 integration time = 0.815 seconds
keister 1 integration time = 0.338 seconds
keister 2 integration time = 0.040 seconds
```

Successful importance sampling makes the transformed integrand, f, more flat. The shorter cubature times correspond to flatter integrands, as illustrated in Fig. 7. The above example uses $d = 1$ to facilitate the plot in Fig. 7; however, the same example works for arbitrary dimensions.

9 Further Work

QMCPy is ripe for growth and development in several areas. We hope that the QMC community will join us in making this a reality.

Multi-level (quasi-)Monte Carlo (ML(Q)MC) methods make possible the computation of expectations of functionals of stochastic differential equations and partial differential equations with random coefficients. Such use cases appear in quantitative finance and geophysical applications. QMCPy's ML(Q)MC's capability is rudimentary, but under active development.

We hope to add a greater variety of use cases and are engaging collaborators to help. Sobol' indices, partial differential equations with random coefficients, expected improvement measures for Bayesian optimization, and multivariate probabilities are some of those on our radar.

Recently, several QMC experts have focused on developing LD generators for Python. Well-established packages such as SciPy [50] and PyTorch [47] are have developed QMC modules that support numerous LD sequences and related functionalities. We plan to integrate the routines as optional backends for QMCPy's LD generators. Creating ties to these other packages will allow users to call their preferred generators from within the QMCPy framework. Moreover, as features in QMCPy become more common and prove their value, we will try to incorporate them into SciPy and other popular, general-purpose packages.

We also plan to expand our library of digital net generating matrices. We wish to incorporate interlaced digital nets, polynomial lattices, and Niederreiter sequences, among others. By including high-quality defaults in QMCPy, we hope to make these sequences more readily available to the public.

Our `DiscreteDistrution` places equal weights on each support point, $\boldsymbol{X}_i$. In the future, we might generalize this to unequal weights.

QMCPy already includes importance sampling, but the choice of sampling distribution must be chosen a priori. We would like to see an automatic, adaptive choice following the developments of [2, 35, 45].

Control variates can be useful for QMC as well as for IID MC [22]. These should be incorporated into QMCPy in a seamless way.

We close with an invitation. Try QMCPy. If you find bugs or missing features, please submit an issue to https://github.com/QMCSoftware/QMCSoftware/issues. If you wish to add your great algorithm or use case, please submit a pull request to our GitHub repository at https://github.com/QMCSoftware/QMCSoftware/pulls. We hope that the community will embrace QMCPy.

Acknowledgements The authors would like to thank the organizers for a wonderful MCQMC 2020. We also thank the referees for their many helpful suggestions. This work is supported in part by SigOpt and National Science Foundation grant DMS-1522687.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/. Software available from tensorflow.org
2. Asmussen, S., Glynn, P.: Stochastic simulation: algorithms and analysis. In: Stochastic Modelling and Applied Probability, vol. 57. Springer, New York (2007). https://doi.org/10.1007/978-0-387-69033-9
3. Bezanson, J., Karpinski, S., Shah, V.B., Edelman, A.: Julia: A fast dynamic language for technical computing (2012). arXiv:1209.5145
4. Burkhardt, J.: Various Software (2020). http://people.sc.fsu.edu/~jburkardt/
5. Choi, S.C.T., Ding, Y., Hickernell, F.J., Jiang, L., Jiménez Rugama, Ll.A., Li, D., Jagadeeswaran, R., Tong, X., Zhang, K., Zhang, Y., Zhou, X.: GAIL: Guaranteed Automatic Integration Library (versions 1.0–2.3.2). MATLAB software. http://gailgithub.github.io/GAIL_Dev/ (2021). https://doi.org/10.5281/zenodo.4018189
6. Choi, S.C.T., Hickernell, F.J., Jagadeeswaran, R., McCourt, M., Sorokin, A.: QMCPy: A quasi-Monte Carlo Python library (2020). https://doi.org/10.5281/zenodo.3964489, https://qmcsoftware.github.io/QMCSoftware/
7. Choi, S.C.T., Hickernell, F.J., Jagadeeswaran, R., McCourt, M.J., Sorokin, A.: QMCPy Documentation (2020). https://qmcpy.readthedocs.io/en/latest/
8. Cools, R., Kuo, F.Y., Nuyens, D.: Constructing embedded lattice rules for multivariate integration. SIAM J. Sci. Comput. **28**(6), 2162–2188 (2006). https://doi.org/10.1137/06065074X
9. Darmon, Y., Godin, M., L'Ecuyer, P., Jemel, A., Marion, P., Munger, D.: LatNet Builder (2018). https://github.com/umontreal-simul/latnetbuilder
10. Dick, J., Kuo, F., Sloan, I.H.: High dimensional integration—the Quasi-Monte Carlo way. Acta Numer. **22**, 133–288 (2013). https://doi.org/10.1017/S0962492913000044
11. Giles, M.: Multi-level (Quasi-)Monte Carlo software (2020). https://people.maths.ox.ac.uk/gilesm/mlmc/
12. Google Inc.: TF Quant Finance: Tensorflow Based Quant Finance Library (2021). https://github.com/google/tf-quant-finance
13. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020). https://doi.org/10.1038/s41586-020-2649-2, https://doi.org/10.1038/s41586-020-2649-2
14. Hickernell, F.J., Choi, S.C.T., Jiang, L., Jiménez Rugama, L.A.: Monte Carlo Simulation, Automatic Stopping Criteria for Wiley StatsRef: Statistics Reference Online, pp. 1–7 (2014)
15. Hickernell, F.J., Sorokin, A.: Quasi-Monte Carlo (QMC) software in QMCPy Google Colaboratory Notebook (2020). http://tinyurl.com/QMCPyTutorial
16. Hickernell, F.J., Sorokin, A.: Quasi-Monte Carlo (QMC) software in QMCPy Google Colaboratory Notebook for MCQMC2020 Article (2020). https://tinyurl.com/QMCPyArticle2021
17. Hickernell, F.J.: A generalized discrepancy and quadrature error bound. Math. Comput. **67**, 299–322 (1998). https://doi.org/10.1090/S0025-5718-98-00894-1
18. Hickernell, F.J.: Goodness-of-fit statistics, discrepancies and robust designs. Statist. Prob. Lett. **44**, 73–78 (1999). https://doi.org/10.1016/S0167-7152(98)00293-4
19. Hickernell, F.J., Jiang, L., Liu, Y., Owen, A.B.: Guaranteed conservative fixed width confidence intervals via Monte Carlo sampling. In: Dick, J., Kuo, F.Y., Peters, G.W., Sloan, I.H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2012, Springer Proceedings in Mathematics and

Statistics, vol. 65, pp. 105–128. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-41095-6
20. Hickernell, F.J., Jiménez Rugama, Ll.A.: Reliable adaptive cubature using digital sequences. In: Cools, R., Nuyens, D. (eds.) Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014, Springer Proceedings in Mathematics and Statistics, vol. 163, pp. 367–383. Springer, Berlin (2016). ArXiv:1410.8615 [math.NA]
21. Hickernell, F.J., Jiménez Rugama, Ll.A., Li, D.: Adaptive quasi-Monte Carlo methods for cubature. In: Dick, J., Kuo, F.Y., Woźniakowski, H. (eds.) Contemporary Computational Mathematics—A Celebration of the 80th Birthday of Ian Sloan, pp. 597–619. Springer, Berlin (2018). https://doi.org/10.1007/978-3-319-72456-0
22. Hickernell, F.J., Lemieux, C., Owen, A.B.: Control variates for quasi-Monte Carlo. Stat. Sci. **20**, 1–31 (2005). https://doi.org/10.1214/088342304000000468
23. Hofert, M., Lemieux, C.: qrng R package (2017). https://cran.r-project.org/web/packages/qrng/qrng.pdf
24. Jagadeeswaran, R., Hickernell, F.J.: Fast automatic Bayesian cubature using lattice sampling. Stat. Comput. **29**, 1215–1229 (2019). https://doi.org/10.1007/s11222-019-09895-9
25. Jagadeeswaran, R., Hickernell, F.J.: Fast automatic Bayesian cubature using Sobol' sampling (2021+). In preparation for submission for publication
26. Jiménez Rugama, Ll.A., Hickernell, F.J.: Adaptive multidimensional integration based on rank-1 lattices. In: Cools, R., Nuyens, D. (eds.) Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014, Springer Proceedings in Mathematics and Statistics, vol. 163, pp. 407–422. Springer, Berlin (2016). ArXiv:1411.1966
27. Keister, B.D.: Multidimensional quadrature algorithms. Comput. Phys. **10**, 119–122 (1996). https://doi.org/10.1063/1.168565
28. Kucherenko, S.: BRODA (2020). https://www.broda.co.uk/index.html
29. Kumaraswamy, P.: A generalized probability density function for double-bounded random processes. J. Hydrol. **46**(1), 79–88 (1980). https://doi.org/10.1016/0022-1694(80)90036-0
30. Kuo, F.Y., Nuyens, D.: Application of quasi-Monte Carlo methods to elliptic PDEs with random diffusion coefficients—a survey of analysis and implementation. Found. Comput. Math. **16**, 1631–1696 (2016). https://people.cs.kuleuven.be/~dirk.nuyens/qmc4pde/
31. Lataniotis, C., Marelli, S., Sudret, B.: Uncertainty quantification in the cloud with UQCloud. In: 4th International Conference on Uncertainty Quantification in Computational Sciences and Engineering (UNCECOMP 2021), pp. 209–217 (2021)
32. L'Ecuyer, P.: SSJ: Stochastic Simulation in Java (2020). https://github.com/umontreal-simul/ssj
33. L'Ecuyer, P., Marion, P., Godin, M., Puchhammer, F.: A tool for custom construction of QMC and RQMC point sets. In: Arnaud, E., Giles, M., Keller, A. (eds.) Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Oxford 2020 (2021+)
34. L'Ecuyer, P., Munger, D.: Algorithm 958: Lattice Builder: a general software tool for constructing rank-1 latice rules. ACM Trans. Math. Softw. **42**, 1–30 (2016)
35. L'Ecuyer, P., Tuffin, B.: Approximate zero-variance simulation. In: Proceedings of the 40th Conference on Winter Simulation, WSC '08, pp. 170–181. Winter Simulation Conference (2008)
36. Marelli, S., Sudret, B.: UQLab: A framework for uncertainty quantification in MATLAB. In: The 2nd International Conference on Vulnerability and Risk Analysis and Management (ICVRAM 2014), pp. 2554–2563. ASCE Library (2014). https://www.uqlab.com
37. Matoušek, J.: On the L_2-discrepancy for anchored boxes. J. Complex. **14**, 527–556 (1998)
38. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia (1992)
39. Nuyens, D.: Magic Point Shop (2017). https://people.cs.kuleuven.be/~dirk.nuyens/qmc-generators/
40. OpenTURNS Developers: An Open Source Initiative for the Treatment of Uncertainties, Risks 'N Statistics (2020). http://www.openturns.org

41. Owen, A.B.: Scrambling Sobol' and Niederreiter-Xing points. J. Complex. **14**(4), 466–489 (1998)
42. Owen, A.B.: On dropping the first Sobol' point. In: Keller, A. (ed.) Monte Carlo and Quasi-Monte Carlo Methods. Springer Proceedings in Mathematics & Statistics, vol. 387, pp. 71–86. Springer, Cham (this volume). https://doi.org/10.1007/978-3-030-98319-2_4
43. Owen, A.B.: Randomized Halton Sequences in R (2020). http://statweb.stanford.edu/~owen/code/
44. Owen, A.B.: About the R function: rsobol (2021). https://statweb.stanford.edu/~owen/reports/seis.pdf
45. Owen, A.B., Zhou, Y.: Safe and effective importance sampling. J. Am. Stat. Assoc. **95**, 135–143 (2000)
46. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
47. PyTorch Developers: PyTorch (2020). https://pytorch.org
48. Robbe, P.: Low Discrepancy Sequences in Julia (2020). https://github.com/PieterjanRobbe/QMC.jl
49. Robbe, P.: Multilevel Monte Carlo simulations in Julia (2021). https://github.com/PieterjanRobbe/MultilevelEstimators.jl
50. SciPy Developers: SciPy Ecosystem (2018). www.scipy.org
51. The MathWorks Inc: MATLAB R2021a. Natick, MA (2020)
52. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nat. Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2

Regular Talks

A Tool for Custom Construction of QMC and RQMC Point Sets

Pierre L'Ecuyer, Pierre Marion, Maxime Godin, and Florian Puchhammer

Abstract We present LatNet Builder, a software tool to find good parameters for lattice rules, polynomial lattice rules, and digital nets in base 2, for quasi-Monte Carlo (QMC) and randomized quasi-Monte Carlo (RQMC) sampling over the s-dimensional unit hypercube. The selection criteria are figures of merit that give different weights to different subsets of coordinates. They are upper bounds on the worst-case error (for QMC) or variance (for RQMC) for integrands rescaled to have a norm of at most one in certain Hilbert spaces of functions. We summarize what are the various Hilbert spaces, discrepancies, types of weights, figures of merit, types of constructions, and search methods supported by LatNet Builder. We briefly discuss its organization and we provide simple illustrations of what it can do.

Keywords Quasi-Monte Carlo · Lattice rules · Polynomial lattice rules · Digital nets · Discrepancy · Software tool

P. L'Ecuyer
Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Montreal, Quebec, Canada
e-mail: lecuyer@iro.umontreal.ca

P. Marion (✉)
Laboratoire de Probabilités, Statistique et Modélisation, LPSM, Sorbonne Université, CNRS, 75005 Paris, France
e-mail: pierre.marion@upmc.fr

M. Godin
Institut des Actuaires, Paris, France
e-mail: maxime.godin@institutdesactuaires.com

F. Puchhammer
Université de Montréal, Montreal, Quebec, Canada
e-mail: fpuchhammer@bcamath.org

Basque Center for Applied Mathematics, Bilbao, Spain

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_3

1 Introduction

QMC methods approximate an integral of the form

$$\mu = \int_0^1 \cdots \int_0^1 f(u_1, \ldots, u_s)\, du_1 \ldots du_s = \int_{(0,1)^s} f(\boldsymbol{u})\, d\boldsymbol{u} = \mathbb{E}[f(\boldsymbol{U})] \tag{1}$$

where $f : (0, 1)^s \to \mathbb{R}$ and $\boldsymbol{U}$ is a uniform random vector over $[0, 1)^s$, by the average

$$\bar{\mu}_n = \frac{1}{n} \sum_{i=0}^{n-1} f(\boldsymbol{u}_i) \tag{2}$$

where $P_n = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}\} \subset [0, 1)^s$ is a set of n deterministic points that cover the unit hypercube more evenly than typical independent random points. That is, the discrepancy between their empirical distribution and the uniform distribution over $[0, 1)^s$ is smaller than for independent random points and converges to 0 faster than $O(n^{-1/2})$ when $n \to \infty$. This discrepancy can be defined in many ways. It usually represents the worst-case integration error for a given class of integrands f. Typically, this class is a reproducing-kernel Hilbert space (RKHS) $\mathcal{H}$ of functions, such that

$$|E_n| := |\bar{\mu}_n - \mu| \le \mathcal{D}(P_n)\mathcal{V}(f) \tag{3}$$

for all $f \in \mathcal{H}$, where $\mathcal{V}(f)$ is the norm of $f - \mu$ in $\mathcal{H}$ (we call it the *variation* of f) and $\mathcal{D}(\cdot)$ is the *discrepancy* measure associated with this Hilbert space [8, 19, 40]. For a fixed $f \in \mathcal{H}$ with $\mathcal{V}(f) > 0$, the error bound in (3) converges at the same rate as $\mathcal{D}(P_n)$. A traditional version of (3), whose derivation does not involve Hilbert spaces, is the classical Koksma-Hlawka inequality, in which $\mathcal{V}(f)$ is the Hardy-Krause variation and $\mathcal{D}(P_n)$ is the star discrepancy, which converges as $O((\log n)^{s-1} n^{-1})$ for well-selected point sets [40]. Another important choice for $\mathcal{H}$ is a Sobolev space of functions whose mixed partial derivatives of order up to α are square-integrable. It is known that for this space, one can construct point sets whose discrepancy converges as $O((\log n)^{(s-1)/2} n^{-\alpha})$, and that this is the best possible rate [4, 8, 15–18]. The main classes of QMC point sets are lattice points and digital nets.

For RQMC, the n QMC points are randomized to provide a set of random points $\{\boldsymbol{U}_0, \ldots, \boldsymbol{U}_{n-1}\} \subset (0, 1)^s$ for which (i) each $\boldsymbol{U}_i$ individually has the uniform distribution over $[0, 1)^s$, and (ii) the points keep their highly-uniform distribution collectively. Randomizations that provably preserve the low discrepancy generally depend on the type of QMC construction: some are used for lattice points and others for digital nets. In some cases, the randomization may even improve the convergence rate of the mean square discrepancy. The RQMC estimator

$$\hat{\mu}_{n,\mathrm{rqmc}} = \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{U}_i), \tag{4}$$

which is now random, is unbiased for μ and one wishes to minimize its variance. For more details on RQMC, see for example [23, 27, 30, 31, 38, 45–47].

The aim of this paper is to introduce *LatNet Builder*, a software tool designed to construct good lattice and digital point sets for QMC and RQMC, in any number of dimensions, for an arbitrary number of points, arbitrary weights on the subsets of coordinates, arbitrary smoothness of the integrands, a variety of construction and randomization methods, and several choices of discrepancies. The point sets can also be extensible in the number of points and number of dimensions. By "constructing the points" here we mean *defining* the set P_n by selecting the parameter values for a general structure, by trying to miminize a *figure of merit* (FOM) that may represent a discrepancy $\mathcal{D}(P_n)$ or be an upper bound on it. Once this is done, other software can be used to randomize and generate the points for their utilization in applications; see [28, 43] for example. LatNet Builder is available in open source at https://github.com/umontreal-simul/latnetbuilder. It is a descendant of *Lattice Builder* [34], whose scope was limited to ordinary lattice rules. Another related tool is Nuyens' fast CBC constructions [41].

The rest of this paper is organized as follows. In Sect. 2, we recall the types of QMC point sets covered by our software, namely ordinary lattice points, polynomial lattice points, digital nets, and their higher-order and interlaced versions, as well as the main randomization methods to turn these point sets into RQMC points. The discrepancies that we consider often provide upper bounds on the mean square integration error when using these randomizations, for certain classes of functions. In Sect. 3, we give the general form of weighted RKHS used in this paper and the corresponding generalized Koksma-Hlawka inequality. We also recall the common types of weights, all supported by the software. In Sect. 4, we review and justify the various discrepancies that are supported by LatNet Builder and can be used as FOMs to select the parameters of point set constructions. In Sect. 5, we summarize the search methods implemented in our software. In Sect. 7, we compare FOM values obtained by various point set constructions and search methods. We also compare RQMC variance for simple integrands f. Section 8 gives a conclusion.

2 Point Set Constructions and Randomizations

LatNet Builder handles ordinary rank-1 lattice points as well as digital nets, which include polynomial lattice rules and high-order and interlaced constructions.

For a *rank-1 lattice rule*, the point set is

$$P_n = \{\boldsymbol{u}_i = i\boldsymbol{v}_1 \bmod 1,\ i = 0, \ldots, n-1\}$$

where $n\boldsymbol{v}_1 = \boldsymbol{a} = (a_1, \ldots, a_s) \in \mathbb{Z}_n^s \equiv \{0, \ldots, n-1\}^s$. It is a *Korobov rule* if $\boldsymbol{a} = (1, a, a^2 \bmod n, \ldots, a^{s-1} \bmod n)$ for some integer $a \in \mathbb{Z}_n$. The parameter to select here is the vector $\boldsymbol{a}$, for any given n. The usual way to turn a lattice rule into an RQMC point set is by a random shift: generate a single random point $\boldsymbol{U}$ uniformly in $(0, 1)^s$, and add it to each point of P_n, modulo 1, coordinate-wise. This satisfies the RQMC conditions. For more details on lattice rules and their randomly-shifted versions, see [20, 21, 27, 30, 33, 50].

The *Digital nets in base 2* handled by LatNet Builder are defined as follows. The number of points is $n = 2^k$ for some integer k. We select an integer $w \geq k$ and s generating matrices $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_s$ of dimensions $w \times k$ and of rank k, with elements in $\mathbb{Z}_2 \equiv \{0, 1\}$. The points $\boldsymbol{u}_i$, $i = 0, \ldots, 2^k - 1$, are defined as follows: for $i = a_{i,0} + a_{i,1}2 + \cdots + a_{i,k-1}2^{k-1}$, we take

$$\begin{pmatrix} u_{i,j,1} \\ \vdots \\ u_{i,j,w} \end{pmatrix} = \boldsymbol{C}_j \begin{pmatrix} a_{i,0} \\ \vdots \\ a_{i,k-1} \end{pmatrix} \bmod 2, \qquad u_{i,j} = \sum_{\ell=1}^{w} u_{i,j,\ell} 2^{-\ell},$$

and $\boldsymbol{u}_i = (u_{i,1}, \ldots, u_{i,s})$. There are more general definitions in [8, 40]. The parameters to optimize are the elements of the matrices $\boldsymbol{C}_j$. Since each $\boldsymbol{C}_j$ has rank k, each one-dimensional projection truncated to its first k digits is $\mathbb{Z}_n/n = \{0, 1/n, \ldots, (n-1)/n\}$. The ordinary digital nets constructed by LatNet Builder often have $w = k$, so the points have only k digits, but this is not always true.

The most popular digital net constructions are still the *Sobol' points* [52], in base $b = 2$, with $k \times k$ generating matrices that are upper triangular and invertible. These matrices are constructed by a specific method, but the bits of the first few columns above the diagonal can be selected arbitrarily, and their choice has an impact on the quality of the net. General-purpose choices have been proposed in [25, 35], e.g., based on the uniformity of two-dimensional projections. LatNet Builder allows one to construct the matrices based on a much more flexible class of criteria.

A *polynomial lattice rule* (PLR) in base 2 with $n = 2^k$ points is defined as follows. We denote by $\mathbb{Z}_2[z]$ the ring of polynomials with coefficients in $\mathbb{Z}_2$, by $\mathbb{L}_2$ the set of formal series of the form $\sum_{\ell=\ell_0}^{\infty} x_\ell z^{-\ell}$ with each $x_\ell \in \mathbb{Z}_2$ and $\ell_0 \in \mathbb{Z}$, and for any given integer $w \geq k$, we define $\varphi_w : \mathbb{L}_2 \to \mathbb{R}$ by

$$\varphi_w \left(\sum_{\ell=\ell_0}^{\infty} x_\ell z^{-\ell} \right) = \sum_{\ell=\max(\ell_0,1)}^{w} x_\ell 2^{-\ell}. \tag{5}$$

We select a *polynomial modulus* $Q = Q(z) \in \mathbb{Z}_2[z]$ of degree k, and a *generating vector* $\boldsymbol{a}(z) = (a_1(z), \ldots, a_s(z)) \in \mathbb{Z}_2[z]^s$, whose coordinates are polynomials of degrees less than k having no common factor with $Q(z)$. The point set of cardinality $n = 2^k$ is

$$P_n = \left\{ \left(\varphi_w \left(\frac{h(z)a_1(z)}{Q(z)} \right), \ldots, \varphi_w \left(\frac{h(z)a_s(z)}{Q(z)} \right) \right) : h(z) \in \mathbb{Z}_2[z], \deg(h(z)) < k \right\}. \tag{6}$$

Here, we want to optimize the vector $\boldsymbol{a}(z)$. This point set turns out to be a digital net in base 2 whose generating matrices $\boldsymbol{C}_j$ contain the first w digits of the binary expansion of the $a_j(z)/Q(z)$. These are Hankel matrices: each row is the previous one shifted to the left by one position, with the last entry determined by the recurrence with characteristic polynomial $Q(z)$, applied to the entries of the previous row. In theory, they have an infinite number of rows, but in practice we truncate them to $w \geq k$ rows. This finite w should be as large as possible to obtain a good approximation of the true PLR points. Typically, $w = 31$, but it could be $w = 63$ if we use 64-bit integers. See [8, 26, 36, 39, 40] for further details on PLRs.

A *high-order polynomial lattice rule* (HOPLR) of order α with $n = 2^k$ points is obtained by constructing an ordinary PLR with polynomial modulus $\tilde{Q}(z)$ of degree αk having $2^{\alpha k}$ points in s dimensions, and using only the first $n = 2^k$ points. See [1, 2, 7]. This type of construction can achieve a higher order of convergence for the error (almost $O(n^{-\alpha})$) than an ordinary PLR for integrands f in a Sobolev space of smoothness order α (i.e., when all mixed partial derivatives of up to order α are square integrable). One drawback is that because of the high degree of $\tilde{Q}$, the cost of a full CBC construction (see Sect. 5) is much higher since there are $2^{\alpha k}$ possibilities to examine each time we select a new coordinate of the generating vector.

Dick [3, 4] also proposed an *interlacing construction*, for digital nets in general (which includes PLRs), that can provide the higher-order convergence rate of almost $O(n^{-\alpha})$ for the integration error, for integrands with smoothness order α. For an interlacing factor $d \in \mathbb{N}$, the method first constructs a digital net in sd dimensions, with generating matrices $C_1, \ldots, C_{sd}$. Then the generating matrices of the s-dimensional interlaced net are $C_1^{(d)}, \ldots, C_s^{(d)}$, where the rows of $C_j^{(d)}$ are the first rows of $C_{(j-1)d+1}, \ldots, C_{jd}$ in this order, then the second rows of these matrices in the same order, and so on.

The simplest way to define a RQMC point set from a digital net in base 2 is to add a digital random shift modulo 2 to all the points. To do this, we generate a single point $\boldsymbol{U} = (U_1, \ldots, U_s)$ uniformly in $(0, 1)^s$, and perform a bitwise exclusive-or (XOR) between the binary digits of $\boldsymbol{U}$ and the corresponding digits of each point $\boldsymbol{u}_i$.

A more involved randomization method for digital nets is the *nested uniform scramble* (NUS) of Owen [45, 46]. In base 2, for each coordinate, we do the following. With probability 1/2, flip the first bit of all the points. Then, for the points whose first bit is 1, with probability 1/2, flip all the second bits. Do the same for the points whose first bit is 0, independently. Then do this recursively for all the bits. After all flipping is done for the first ℓ bits, partition the points in 2^ℓ batches according to the values of their first ℓ bits, and for each batch, flip bit $\ell + 1$ of all the points with probability 1/2, independently across the batches. This requires $(2^\ell - 1)s$ random bits to flip the first ℓ bits of all coordinates. One can equivalently do this only for the first k bits, and generate the other bits randomly and independently across points [38].

A less expensive scramble, which gives less independence than NUS but more than a digital random shift, is a (left) *linear matrix scramble* (LMS) followed by a digital random shift (LMS+shift) [23, 24, 38, 48]. The LMS replaces $\boldsymbol{C}_j$ by $\tilde{\boldsymbol{C}}_j = \boldsymbol{L}_j \boldsymbol{C}_j \bmod 2$, where $\boldsymbol{L}_j$ is a random non-singular lower-triangular $w \times w$ binary matrix.

Owen [46] proved that under sufficient smoothness conditions on f, the RQMC variance with NUS on digital nets with fixed s and bounded t converges as $O(n^{-3}(\log n)^{s-1})$. A variance bound of the same order was shown for LMS+shift in [23, 54]. Note that these results were proved under the assumption that $w = \infty$.

3 Hilbert Spaces and Projection-Dependent Weights

The FOMs used by LatNet Builder are based on generalized (weighted) Koksma-Hlawka inequalities of the form (3) where

$$\mathcal{V}^p(f) = \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,2,\ldots,s\}} \gamma_{\mathfrak{u}}^{-p} \mathcal{V}^p(f_{\mathfrak{u}}) \tag{7}$$

and

$$\mathcal{D}^q(P_n) = \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1,2,\ldots,s\}} \gamma_{\mathfrak{u}}^{q} \mathcal{D}_{\mathfrak{u}}^q(P_n), \tag{8}$$

where $1/p + 1/q = 1$, $\gamma_{\mathfrak{u}} \in \mathbb{R}$ is a weight assigned to the subset $\mathfrak{u}$, $\mathcal{V}(f_{\mathfrak{u}})$ is the variation of $f_{\mathfrak{u}}$, $\mathcal{D}_{\mathfrak{u}}(P_n)$ is the discrepancy of the projection of P_n over the subset $\mathfrak{u}$ of coordinates, and $f = \sum_{\mathfrak{u} \subseteq \{1,2,\ldots,s\}} f_{\mathfrak{u}}$ is the functional ANOVA decomposition of f [10, 47]. LatNet Builder allows any $q \in [1, \infty]$. Taking $q = \infty$ with $p = 1$ means removing the q and taking the max instead of the sum in (8), while $p = \infty$ with $q = 1$ means removing the p and taking the max instead of the sum in (7). The most common choice is $p = q = 2$.

LatNet Builder implements a variety of choices for $\mathcal{D}_{\mathfrak{u}}(P_n)$, depending on the point set constructions. Some of these measures correspond to the worst-case error in some function space, assuming that the points of P_n are not randomized. Others correspond to the mean-square error (or variance), assuming that the points are randomized in some particular way. This is typically done by defining a RKHS with a kernel that is invariant with respect to the given randomization (i.e., digital shift-invariant, scramble-invariant, etc.), and taking the worst-case error in that space.

The role of the weights is to better recognize the importance of the subsets $\mathfrak{u}$ for which $f_{\mathfrak{u}}$ contributes the most to the error or variance. That is, if $\mathcal{V}(f_{\mathfrak{u}})$ is unusually large, we want to divide it by a larger weight $\gamma_{\mathfrak{u}}$ to control its contribution to $\mathcal{V}(f)$, but then we have to multiply $\mathcal{D}_{\mathfrak{u}}(P_n)$ in (8) by the same weight. The final effect is that the FOM will penalize more the discrepancy for that particular projection.

In principle, the weights $\gamma_{\mathfrak{u}}$ can be arbitrary. But for large s, defining arbitrary individual weights for the $2^s - 1$ projections is impractical, so special forms of weights that are parameterized by much fewer than $2^s - 1$ parameters have been proposed. The most common ones are *product weights*, for which a weight γ_j is assigned to coordinate j for $j = 1, \ldots, s$, and $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$; *order-dependent weights*, for which $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|}$ where $\Gamma_1, \ldots, \Gamma_s$ are selected constants and $|\mathfrak{u}|$ is the cardinality of of $\mathfrak{u}$; and the *product-and-order-dependent (POD) weights*, which are a combination

of the two, with $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j\in\mathfrak{u}} \gamma_j$. These are all available in LatNet Builder. For more discussion on how to select the weights, see [8, 12, 32–34], for example.

LatNet Builder can construct point sets that are extensible in the number of dimensions and also in the number of points, which means that we can construct point sets that perform well in the first s dimensions for $s = s_{\min}, \ldots, s_{\max}$, and/or if we take the first n points for $n = n_1, n_2, \ldots, n_m$, simultaneously. Typically, one would take $n_j = 2^{k_{\min}+j-1}$ for $j = 1, \ldots, m$, so $n_m = 2^{k_{\max}} = 2^{k_{\min}+m-1}$ [22]. The global FOM in this case will be a weighted sum or maximum of the FOMs over the considered dimensions s and/or cardinalities n_j. The CBC construction approach described in Sect. 5 already gives a way to implement the extension in s. For the extension in n (or k), LatNet Builder implements criteria and heuristic search methods that account for a global FOM.

4 Figures of Merit

In this section, we review the FOMs implemented in LatNet Builder. Most of them have the general form (8) where typically, when the points have the appropriate special structure of a lattice, polynomial lattice, or digital net, and with an adapted FOM, we have

$$\mathcal{D}^q_{\mathfrak{u}}(P_n) = \frac{1}{n} \sum_{i=0}^{n-1} \prod_{j\in\mathfrak{u}} \phi(u_{i,j}) \tag{9}$$

for some function $\phi : [0, 1) \to \mathbb{R}$. With product weights $\gamma_{\mathfrak{u}} = \prod_{j\in\mathfrak{u}} \gamma_j$, this becomes

$$\mathcal{D}^q(P_n) = -1 + \frac{1}{n} \sum_{i=0}^{n-1} \prod_{j=1}^{s} (1 + \gamma_j^q \phi(u_{i,j})),$$

which can be computed with $O(ns)$ evaluations of ϕ.

As an illustration, for a randomly-shifted lattice rule, the variance is:

$$\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \sum_{\mathbf{0}\neq\boldsymbol{h}\in L_s^*} |\hat{f}(\boldsymbol{h})|^2, \tag{10}$$

where $L_s^* \subset \mathbb{Z}^s$ is the *dual lattice* [30]. It is also known that for periodic continuous functions having square-integrable mixed partial derivatives up to order $\alpha/2$ for an even integer $\alpha \geq 2$, one has $|\hat{f}(\boldsymbol{h})|^2 = O((\max(1, h_1) \cdots \max(1, h_s))^{-\alpha})$. This motivates the well-known FOM [33, 40, 50]:

$$\mathcal{P}_\alpha := \sum_{\mathbf{0}\neq \boldsymbol{h}\in L_s^*} (\max(1,h_1)\dots\max(1,h_s))^{-\alpha}$$
$$= \frac{1}{n}\sum_{i=0}^{n-1} \sum_{\emptyset\neq \mathfrak{u}\subseteq\{1,\dots,s\}} \left(\frac{-(-4\pi^2)^{\alpha/2}}{\alpha!}\right)^{|\mathfrak{u}|} \prod_{j\in\mathfrak{u}} B_\alpha(u_{i,j}) \tag{11}$$

where $B_{\alpha/2}$ is the Bernoulli polynomial of degree $\alpha/2$ ($B_1(u) = u - 1/2$, $B_2(u) = u^2 - u + 1/6$, etc.), and the equality in (11) holds only when α is an even integer. Moreover, there are rank-1 lattices point sets P_n for which $\mathcal{P}_\alpha$ converges as $O(n^{-\alpha+\epsilon})$ for any $\epsilon > 0$ [9, 49, 50]. Adding projection-dependent weights $\gamma_\mathfrak{u}$ leads to the weighted $\mathcal{P}_{\gamma,\alpha}$, defined by (8) and (9) with $q = 2$,

$$\phi(u_{i,j}) = -(-4\pi^2)^{\alpha/2} B_\alpha(u_{i,j})/\alpha!,$$

and $\mathcal{D}^2_\mathfrak{u}(P_n) = \mathcal{P}_{\alpha,\mathfrak{u}}(P_n)$ is the $\mathcal{P}_\alpha$ for the projection of P_n on the coordinates in $\mathfrak{u}$.

There is a similar variance expression for digital nets in base 2 with a random digital shift, with the Fourier coefficients $\hat{f}(\boldsymbol{h})$ replaced the the Walsh coefficients, and the dual lattice replaced by the dual net [8, Definition 4.76] or the dual lattice in the case of PLRs [26, 36]. Thus, FOMs that correspond to variance bounds can be obtained by finding easily computable bounds on the Walsh coefficients. By assuming a rate of decrease of $O(2^{-\alpha|\boldsymbol{h}|})$ of the Walsh coefficients $\tilde{f}(\boldsymbol{h})$ with $\boldsymbol{h} = (h_1,\dots,h_s) \in \mathbb{N}^s$ and $|\boldsymbol{h}| = |h_1| + \cdots + |h_s|$, and using a RKHS with shift- and scramble-invariant kernel, [6, 54] obtain a FOM of the form (8) and (9) with

$$\phi(x) = \phi_\alpha(x) = \mu(\alpha) - \mathbb{I}[x > 0]\cdot 2^{(1+\lfloor \log_2(x)\rfloor)(\alpha-1)}(\mu(\alpha)+1),$$

where $\mathbb{I}$ is the indicator function, $-\lfloor \log_2 x\rfloor$ is the index of the first nonzero digit in the expansion of x, and $\mu(\alpha) = \left(1 - 2^{1-\alpha}\right)^{-1}$ for any real number $\alpha > 1$. This gives $\mu(2) = 2$, $\mu(3) = 4/3$,…, For $\alpha = 2$, this gives

$$\phi_2(x) = 2(1 - \mathbb{I}[x > 0]\cdot 3\cdot 2^{\lfloor \log_2(x)\rfloor}),$$

which corresponds to the FOM suggested in [36, Sect. 6.3] for PLRs. In [23, 54], $\phi(x)$ is written in terms of $\eta = \alpha - 1$ instead, but it is exactly equivalent. These papers also show the existence of digital nets for which the FOM converges as $O(n^{-\alpha}(\log n)^{s-1})$ for any $\alpha > 1$. This FOM can be seen as a counterpart of $\mathcal{P}_\alpha$ and we call it $\tilde{\mathcal{P}}_\alpha$. Its value $\tilde{\mathcal{P}}_{\alpha,\mathfrak{u}}(P_n)$ on the projection of P_n on the coordinates in $\mathfrak{u}$ can be used for $\mathcal{D}^2_\mathfrak{u}(P_n)$, with $q = 2$. Note that under our assumption that the first k rows of each generating matrix are linearly independent, $-\lfloor \log_2(u_{i,j})\rfloor$ never exceeds k when $u_{i,j} \neq 0$, and therefore this FOM depends only on the first k bits of output.

Dick and Pillichshammer [8, Chap. 12] consider a RKHS with shift-invariant kernel, which is a weighted Sobolev space of functions whose mixed partial derivatives of order 1 are square-integrable. This gives a FOM of the form (8) and (9) with $q = 2$ and

$$\phi(x) = 1/6 - \mathbb{I}[x > 0] \cdot 2^{\lfloor \log_2(x) \rfloor - 1}.$$

They show that there are digital nets for which this FOM (and therefore the square error) converges almost as $O(n^{-2})$. In their Chap. 13, they find that the scramble-invariant version gives the same ϕ. Note that this $\phi(x)$ is equal to $\phi_2(x)$ above, divided by 12. Therefore, we can get the corresponding FOM just from $\tilde{\mathcal{P}}_2$ by multiplying the weights by order-dependent factors of $1/12^j$ for order j.

Goda [13] examines *interlaced polynomial lattice rules* (IPLR), also for a Sobolev space of order α, with an interlacing factor $d > 1$. He provides two upper bounds on the worst-case error in a deterministic setting. These bounds can be used as FOMs. The first is valid for all positive integer values of α and $d > 1$, whereas the second holds only for $1 < d \le \alpha$, but is tighter when it applies. These two bounds have the form (8) and (9) with $q = 1$, $\gamma_{\mathfrak{u}}$ replaced by $\tilde{\gamma}_{\mathfrak{u}}$, and

$$\phi(x_{i,j}) = -1 + \prod_{\ell=1}^{d} (1 + \phi_{\alpha,d,\ell}(x_{i,(j-1)d+\ell})),$$

where for the first bound, $\tilde{\gamma}_{\mathfrak{u}} = \gamma_{\mathfrak{u}} 2^{\alpha(2d-1)|\mathfrak{u}|/2}$,

$$\phi_{\alpha,d,\ell}(x) = \frac{1 - 2^{(\min(\alpha,d)-1)\lfloor \log_2 x \rfloor}(2^{\min(\alpha,d)} - 1)}{2^{(\alpha+2)/2}(2^{\min(\alpha,d)-1} - 1)}$$

for all $x \in [0, 1)$, where $2^{\lfloor \log_2 0 \rfloor} = 0$, while for the second bound, $\tilde{\gamma}_{\mathfrak{u}} = \gamma_{\mathfrak{u}}$ and

$$\phi_{\alpha,d,\ell}(x) = \frac{2^{d-1}(1 - 2^{(d-1)\lfloor \log_2 x \rfloor}(2^d - 1))}{2^{\ell}(2^{d-1} - 1)}.$$

One can achieve a convergence rate of almost $O(n^{-\min(\alpha,d)})$ for these FOMs (and therefore for the worst-case error). We denote these two FOMs by $\mathcal{I}^{(a)}_{\alpha,d}$ and $\mathcal{I}^{(b)}_{\alpha,d}$.

Goda and Dick [14] proposed another FOM, also for a Sobolev space of order α, for interlaced randomly-scrambled PLRs of high order. They showed that this scheme can achieve the best possible convergence rate of $O(n^{-(2\min(\alpha,d)+1)+\delta})$ for the variance. The FOM, denoted $\mathcal{I}^{(c)}_{\alpha,d}$, has the same form, but with $q = 2$,

$$\phi(x) = \phi_{\alpha,d}(x) = \frac{1 - 2^{2\min(\alpha,d)\lfloor \log_2 x \rfloor}(2^{2\min(\alpha,d)+1} - 1)}{2^{\alpha}(2^{2\min(\alpha,d)} - 1)},$$

and $\gamma_{\mathfrak{u}}$ replaced by $\tilde{\gamma}_{\mathfrak{u}} = \gamma_{\mathfrak{u}} D_{\alpha,d}^{|\mathfrak{u}|}$ where $D_{\alpha,d} = 2^{2\max(d-\alpha,0)+(2d-1)\alpha}$.

Another set of FOMs are obtained from upper bounds on the star discrepancy of $\mathcal{D}^*(P_n)$ or its projections on subsets of coordinates, when P_n is a digital (t, k, s)-net. One such bound is $\mathcal{D}^*(P_n) \le 1 - (1 - 1/n)^s + \mathcal{R}_2$ where

$$\mathcal{R}_2 = -1 + \frac{1}{n}\sum_{i=0}^{n-1}\prod_{j=1}^{s}\left[\sum_{k=0}^{n-1} 2^{-\lfloor \log_2 k\rfloor - 1}\mathrm{wal}_k(u_{i,j})\right] \tag{12}$$

wal_k is the kth Walsh function in one dimension, and we assume that the generating matrices $\boldsymbol{C}_j$ are $k \times k$. See [8, Theorems 5.34 and 5.36], where a more general version with projection-dependent weights is also given. For PLRs in base $b = 2$, this criterion is equal to $\mathcal{R}'_{2,\gamma}$ given in [8, Chap. 10]:

$$\mathcal{R}'_{2,\gamma} = -\sum_{\emptyset\neq\mathfrak{u}\subseteq\{1,2,\ldots,s\}}\gamma_{\mathfrak{u}} + \frac{1}{n}\sum_{i=0}^{n-1}\sum_{\emptyset\neq\mathfrak{u}\subseteq\{1,2,\ldots,s\}}\gamma_{\mathfrak{u}}\prod_{j\in\mathfrak{u}}\phi_k(u_{i,j}) \tag{13}$$

where $\phi_k(u) = -\lfloor \log_2(u)\rfloor/2$ if $u \geq 2^{-k}$ and $\phi_k(u) = 1 + k/2$ otherwise.

A classical upper bound on the star discrepancy is also given by the t-value of the digital net:

$$\mathcal{D}^*(P_n) \leq \frac{2^t}{n}\sum_{j=0}^{s-1}\binom{k-t}{j}.$$

If we use this upper bound for each projection on the subset $\mathfrak{u}$ of coordinates, we get the FOM (8) with $q = 1$ and

$$\mathcal{D}_{\mathfrak{u}}(P_n) = \frac{2^{t_{\mathfrak{u}}}}{n}\sum_{j=0}^{|\mathfrak{u}|-1}\binom{k-t_{\mathfrak{u}}}{j}$$

where $t_{\mathfrak{u}} = t_{\mathfrak{u}}(P_n)$ is the t-value of the projection of P_n on the coordinates in $\mathfrak{u}$. LatNet Builder implements this with arbitrary weights, using algorithms described in [37]. Dick [4] obtains worst-case error bounds that converge at rate almost $O(n^{-\alpha})$ for interlaced digital nets, based on the t-values of the projections.

5 Search Methods

For given construction type, FOM, and weights, finding the best choice of parameters may require to try all possibilities, but their number is usually much too large. LatNet Builder implements the following search methods.

In an *exhaustive search*, all choices of parameters are tried, so we are guaranteed to find the best one. This is possible only when there are not too many possibilities.

A *random search* samples uniformly and independently a fixed number r of parameter choices, and the best one is retained.

In a *full component-by-component (CBC)* construction, the parameters are selected for one coordinate at a time, by taking into account the choices for the previous coordinates only [5, 8, 51]. The parameters for coordinate j (e.g., the jth coordinate of

the generating vector in the case of lattices), are selected by minimizing the FOM for the first j coordinates, in j dimensions, by examining all possibilities of parameters for this coordinate, without changing the parameter choices for the previous coordinates. This is done for the s coordinates in succession. This greedy approach can reduce by a huge factor (exponential in the dimension) the total number of cases that are examined in comparison with the exhaustive search. What is very interesting is that for most types of QMC constructions and FOMs implemented in LatNet Builder, the convergence rate for the worst-case error or variance obtained with this restricted approach is provably the same as for the exhaustive search [8, 15].

For lattice-type point sets, with certain FOMs and choices of weights (e.g., $\mathcal{P}_2$ and $\tilde{\mathcal{P}}_2$ with product and/or order-dependent weights), a *fast CBC* construction can be implemented by using a fast Fourier transform (FFT), so the full CBC construction can be performed much faster [8, 42, 44]. LatNet Builder supports this.

When the number of choices for each coordinate is too large or fast-CBC does not apply, one can examine only a fixed number of random choices for each coordinate j; we call this the *random CBC* construction.

For lattice-type constructions, one can also further restrict the search to *Korobov*-type generating vectors. The first coordinate is set to 1 and only the second coordinate needs to be selected. This can be done either by an exhaustive search or by just taking a random sample for the second coordinate (*random Korobov*).

For digital nets, a *mixed CBC* method is also available: it uses full CBC for the first $d - 1$ coordinates and random CBC for the other ones, for given $1 \le d \le s$.

6 Usage of the Software Tool

At the first level, LatNet Builder is a library written in C++ which implements classes and methods to compute FOMs and search for good point sets for all the construction methods and FOMs discussed in this paper. The source code and a detailed reference manual for the library are provided at https://github.com/umontreal-simul/latnetbuilder. The library can be used directly from C++ programs and can be extended if desired. This is the most flexible option, but it requires knowledge of C++ and the library.

At the second level, there is an executable `latnetbuilder` program that can be called directly from the command line in Linux, Mac OS, or Windows. This program has a large number of options to specify the type and number of points, number of dimensions, search method, FOM, weights, output file format, etc. We think this is the most convenient way of using LatNet Builder in practice. In addition to giving the search results on the terminal, the program creates a directory with two output files: one summarizes the search parameters and the other one puts the parameters of the selected point set in a standard format designed for reading by other software that can generate and use the RQMC points in applications. There are selected file formats for ordinary lattice rules, polynomial lattice rules, Sobol points, and general

digital nets in base 2. A tutorial on the command line and a summary of the options can be found in the reference manual.

At a third level, there is a Java interface in SSJ, a Python interface included in the distribution, and a Graphical User Interface (GUI) based on the Jupyter ecosystem, written in Python. These interfaces use the command line internally. With the GUI, the user can select the desired options in menus, write numerical values in input cells, write the name of the desired output directory, and launch a search. The LatNet Builder program with the Python interface and the GUI can be installed as a Docker container on one's machine. An even simpler access to the GUI is available without installing anything: just click on the "Launch Binder" black and pink link on the GitHub site and it will run the GUI with a version of the program hosted by Binder. This service provides limited computation resources but is convenient for small experiments and to get a sense of what the software does.

We now give examples to illustrate how the command line works, how the results look like, and give some idea of the required CPU times for the search. The timings were made in a VirtualBox for Ubuntu Linux running atop Windows 10 on an old desktop computer with an Intel i7-2600 processor at 3.4GHz and 32 Gb of memory.

The following command makes a search for a polynomial lattice rule with $n = 2^{16}$ points in 256 dimensions, with the default irreducible polynomial modulus, using the fast-CBC search method, the $\tilde{P}_2$ criterion, $q = 2$, and order-dependent weights $\Gamma_2 = 10$, $\Gamma_3 = 0.1$, $\Gamma_4 = 0.001$, and the other weights equal to 0 (recall that Γ_1 has no impact on the selection). These weights decrease quickly with the order because the number of projections of any given order increases very quickly with the order. If they decrease too slowly, the total weight of the projections of order 2 in the FOM will be negligible compared to those of order 4, for instance. For a smaller s, the weights may decrease more slowly.

```
latnetbuilder -t lattice -c polynomial -s 2^16 -d 256 -e fast-CBC
-f CU:P2 -q 2 -w order-dependent:0,0,10.,0.1,0.001 -O lattice
```

This search takes about 840 s to complete and the retained rule had an FOM of 60.235. About 70% of this FOM is contributed by the projections of order 4 and less than 1% by the projections of order 2. The output file looks like:

```
# Parameters for a polynomial lattice rule in base 2
256     # s = 256 dimensions
16      # n = 2^16 = 65536 points
96129   # polynomial modulus
1       # coordinates of generating vector, starting at j=1
47856
60210
44979
⋮
```

Instead of making the search directly in the polynomial lattice space, we can make the same search by viewing the PLR as a digital net, using the option "`-t net`." With that option, fast-CBC search is not available, but we can do a random CBC search, say with 100 samples for each coordinate. With either search method, instead of reporting the modulus and generating vector of the PLR in the output file,

we can have all the columns of the generating matrices, by using the option "`-O net`" instead of "`-O lattice`." The command is:

```
latnetbuilder -t net -c polynomial -s 2^16 -d 256
-e random-CBC:100 -f CU:P2 -q 2
-w order-dependent:0,0,10.,0.1,0.001 -O net
```

With this, the search takes about 160 s and gives an FOM of 63.25. The output file looks like this, with 16 integers per dimension, one integer for each column:

```
# Parameters for a digital net in base 2
256  # s = 256 dimensions
16   # n = 2^16 = 65536 points
31   # r = 31 binary output digits
# Columns of gen. matrices C_1,...,C_s, one matrix per line:
33260 66520 133040 266081 532162 1064325 2128651 4257303  ...
1561357389 975231131 1950462262 1753440876 1359398105 ...
1642040599 1136597551 125711455 251422911 502845823 ...
⋮
```

For a search in 32 dimensions instead of 256, it takes about 40 s for the first case and 8 s for the second case.

As another example, the following command launches a search for good direction numbers for Sobol' points for up to 2^{16} points in 256 dimensions. It uses a mixed CBC search which does a full CBC evaluation for the first 10 coordinates and then a random-CBC search with 100 random samples for each of the remaining coordinates. The criterion is the maximum t-value with order-dependent weights of $\Gamma_2 = 1.0$, $\Gamma_1 = 0.5$, and 0 for everything else. Here, since we take the sup over the projections, the weights can decrease much more slowly.

```
latnetbuilder -t net -c sobol -s 2^16 -d 256 -e mixed-CBC:100:10
-f projdep:t-value -q inf -w order-dependent:0:0,1.0,0.5
```

The search took about 3340 s and returned a FOM of 8.0. The output file provides the retained direction numbers and it looks like this:

```
# Initial direction numbers m_{j,c} for Sobol points
# s = 256 dimensions
1    # This is m_{j,k} for the second coordinate
1 1
1 1 1
1 1 1
1 1 5 1
1 3 5 1
1 1 1 9 9
1 1 1 9 17
⋮
```

If we change s to 32, the search takes 12 s and the FOM is 5.0 instead.

7 Simple Numerical Illustrations

Here we give a few simple examples of what LatNet Builder can do. The simulation experiments, including the generation and randomization of the points, were done using SSJ [29].

7.1 FOM Quantiles for Different Constructions

One might be interested in estimating the probability distribution of FOM values obtained when selecting parameters at random for a given type of construction, perhaps under some constraints, and as a function of n. Here we estimate this distribution by its empirical counterpart with an independent sample of size 1000 (with replacement), and we report the 0.1, 0.5 and 0.9 quantiles of this empirical distribution, for n going from 2^6 to 2^{18}. We do this for PLRs, Sobol' points, and digital nets with arbitrary invertible and projection-regular generating matrices (random nets), with $\tilde{\mathcal{P}}_2$ taken as the FOM, in $s = 6$ dimensions, with $\gamma_{\mathfrak{u}} = 0.7^{|\mathfrak{u}|}$ for all $\mathfrak{u}$. We also report the value obtained by a (full) fast CBC search for a PLR. The results are displayed in the first panel of Fig. 1. We see that the FOM distribution has a smaller mean and much less variance for the Sobol' points than for the other contructions. Even the median obtained for Sobol' beats (slightly) the FOM obtained by a full CBC construction with PLRs. The quantiles for random PLRs and random nets are approximately the same.

The second panel of the figure shows the results of a similar sampling for PLRs with $\mathcal{I}^{(c)}_{2,2}$ as a FOM, also in 6 dimensions. Here, the FOM values are more dispersed and the fast CBC gives a significantly better value than the best FOM obtained by random sampling. Also the search for the point set parameters is much quicker with the fast CBC construction than with random sampling of size 1000.

7.2 Comparison with Tabulated Parameter Selections

We now give small examples showing how searching for custom parameter values with LatNet Builder can make a difference in the RQMC variance compared with pre-tabulated parameter values available in software or over the Internet. We do this for Sobol' nets, and our comparison is with the precomputed direction numbers obtained by Joe and Kuo [25], which are arguably the best proposed values so far. These parameters were obtained by optimizing a FOM based on the t-values over two-dimensional projections, using a CBC construction. With LatNet Builder, we can account for any selected projections in our FOM. For instance, if we think all the projections in two and three dimensions are important, we can select a FOM that accounts for all these projections. To illustrate this, we made a CBC construction of

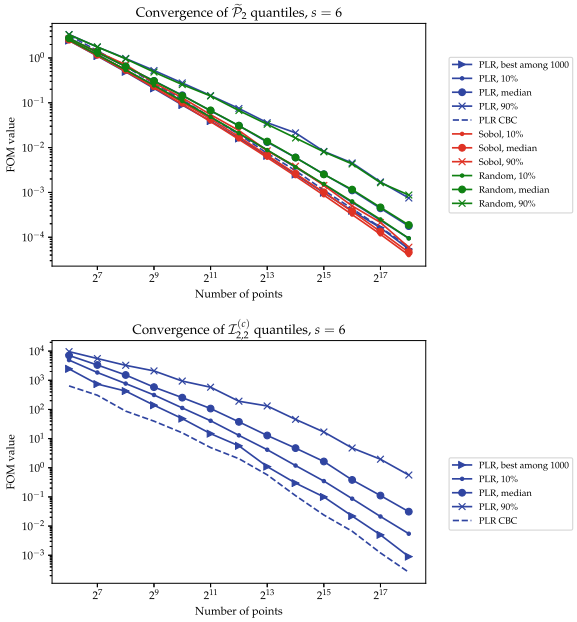


Fig. 1 The 0.1, 0.5, and 0.9 quantiles of the FOM distribution as functions of n for various constructions, in log-log scale

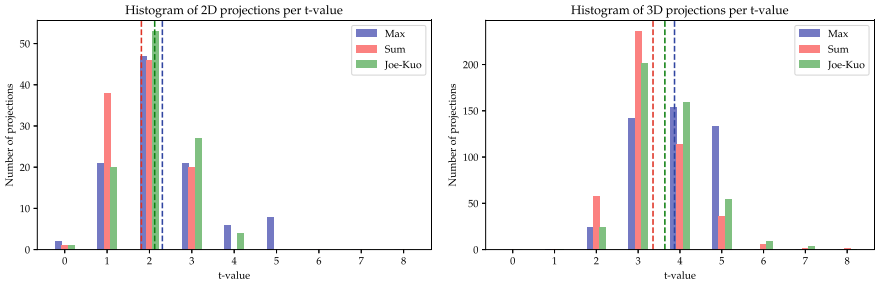


Fig. 2 Distributions of t-values for 2-dim and 3-dim projections, for three Sobol' point sets: (1) *Joe-Kuo* taken from [25], (2) *Max* and (3) *Sum* are found by LatNet Builder as explained in the text. For each case, we report the number of projections having any given t-value, as well as the average t-value (dashed vertical lines)

$n = 2^{12}$ Sobol' points in $s = 15$ dimensions, using the sum or the maximum of t-values in the two- and three-dimensional projections. Figure 2 shows the distribution of t-values obtained with the sum, the max, and the points from [25]. Compared with the latter, we are able to reduce the worse t-value over 3-dim projections from 8 to 5 when using the max, and to reduce the average t-value when using the sum. However, when using the max, we get a few poor two-dim projections, because we compare the t-values on the same scale for two and three dimensions. We should probably multiply the t-value by a scaling factor that decreases with the dimension.

In our next illustration, we compare the RQMC variances for Sobol' points with direction numbers taken from [25] and direction numbers found by LatNet Builder using a custom FOM for our function. We want to integrate

$$f(\boldsymbol{u}) = \prod_{j=1}^{5}(\psi(u_j) - \mu) + \prod_{j=6}^{10}(\psi(u_j) - \mu),$$

where $\psi(u) = \left((u - 0.5)^2 + 0.05\right)^{-1}$ and $\mu = \mathbb{E}[\psi(U)] \approx 10.3$ when $U \sim U(0, 1)$. This function is the sum of two five-dimensional ANOVA terms for a more general function taken from [11]. A good FOM for this function should focus mainly on these two five-dim projections, namely $\mathfrak{u} = \{1, 2, 3, 4, 5\}$ and $\mathfrak{u} = \{6, 7, 8, 9, 10\}$, and not on the two-dim projections as in [25]. So we made a search with the $\tilde{\mathcal{P}}_2$ criterion with weights $\gamma_{\mathfrak{u}} = 1$ for these two projections and 0 elsewhere, to obtain new direction numbers for $n = 2^{20}$ Sobol' points in 10 dimensions. Then we estimated the variance of the sample RQMC average over these n points with the two choices of direction numbers (those of [25] and ours), using $m = 200$ independent replications of an RQMC scheme that used only a random digital shift. The empirical variance with our custom points was smaller by a factor of more than 18.

7.3 *Variance for Another Toy Function*

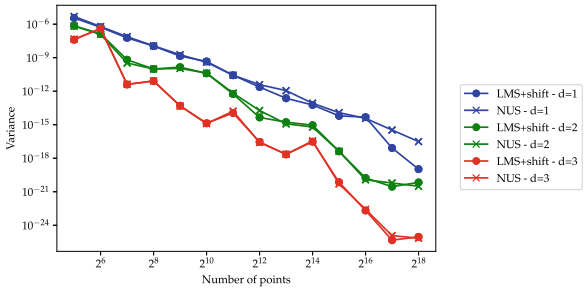
Here we consider a family of test functions similar to those in [53]:

$$f_{s,\boldsymbol{c}}(\boldsymbol{u}) = \prod_{j=1}^{s}(1 + c_j \cdot (u_j - 1/2))$$

for $\boldsymbol{u} \in (0, 1)^s$, where $\boldsymbol{c} = (c_1, \ldots, c_s) \in (0, 1)^s$. The ANOVA components are, for all $\mathfrak{u} \subset \{1, \ldots, s\}$,

$$(f_{s,\boldsymbol{c}})_{\mathfrak{u}}(\boldsymbol{u}) = \prod_{j \in \mathfrak{u}} c_j \cdot (u_j - 1/2)).$$

For an experiment, we take arbitrarly $s = 3$ and $\boldsymbol{c} = (0.7, 0.2, 0.5)$. We use LatNet Builder to search for good PLRs with a fast CBC construction, with product weights $\gamma_j = c_j$, with the FOMs $\mathcal{P}_2$, $\mathcal{I}_{2,2}^{(c)}$, and $\mathcal{I}_{3,3}^{(c)}$ (whose interlacing factors d are 1, 2, and 3, respectively). For each $n = 2^k$, $k = 5, \ldots, 18$, we estimate the RQMC variance with m independent replications of the randomization scheme, with $m = 1000$ for LMS+shift, and $m = 100$ for NUS. For the interlaced points, the randomization is performed before the interlacing, as in [14]. Figure 3 shows the variance as a function of n, in log-log scale. We see that the two randomization schemes give approximately the same variance. However, the time to generate and randomize the points is much larger for NUS than for LMS+shift: around 10 times longer for 2^{11} points and 50



Nb. of points	Constr. time
128	0.02 s
512	0.04 s
2048	0.12 s
8192	0.50 s
32768	2.25 s
131072	8.32 s

Fig. 3 Variance as a function of n in log-log scale, for PLRs with two randomization schemes and three interlacing factors d, found with LatNet Builder. We also report the average time to generate and randomize the points with LMS+shift

times longer for 2^{18} points. As expected, the variance reduction and the convergence rate are larger when the interlacing factor increases, although the curves are more noisy.

8 Conclusion

LatNet Builder is both a tool for researchers to study the properties of highly uniform point sets and associated figures of merit, and for practitioners who want to find good parameters for a specific task. It is relatively easy to incorporate new FOMs into the software, especially if they are in the kernel form (9).

Many questions remain open regarding the roles of the construction, the search method, the randomization, and (perhaps more importantly) the choice of the weights. It is our hope that the software presented here will spur interest into these issues.

Acknowledgements This work has been supported by a NSERC Discovery Grant and an IVADO Grant to P. L'Ecuyer, and by a stipend from Corps des Mines to P. Marion. F. Puchhammer was supported by Spanish and Basque governments fundings through BCAM (ERDF, ESF, SEV-2017-0718, PID2019-108111RB-I00, PID2019-104927GB-C22, BERC 2018e2021, EXP. 2019/00432, KK-2020/00049), and the computing infrastructure of i2BASQUE and IZO-SGI SGIker (UPV). Yocheved Darmon wrote code for producing the output files.

References

1. Baldeaux, J., Dick, J., Greslehner, J., Pillichshammer, F.: Construction algorithms for higher order polynomial lattice rules. J. Complex. **27**, 281–299 (2011)
2. Baldeaux, J., Dick, J., Leobacher, G., Nuyens, D., Pillichshammer, F.: Efficient calculation of the worst-case error and (fast) component-by-component construction of higher order polynomial lattice rules. Numer. Algorithms **59**(3), 403–431 (2012)

3. Dick, J.: Explicit constructions of Quasi-Monte Carlo rules for the numerical integration of high-dimensional periodic functions. SIAM J. Numer. Anal. **45**(5), 2141–2176 (2007)
4. Dick, J.: Walsh spaces containing smooth functions and Quasi-Monte Carlo rules of arbitrary high order. SIAM J. Numer. Anal. **46**(3), 1519–1553 (2008)
5. Dick, J., Kuo, F.Y., Pillichshammer, F., Sloan, I.H.: Construction algorithms for polynomial lattice rules for multivariate integration. Math. Comput. **74**(248), 1895–1921 (2005)
6. Dick, J., Pillichshammer, F.: Multivariate integration in weighted Hilbert spaces based on Walsh functions and weighted Sobolev spaces. J. Complex. **21**(2), 149–195 (2005)
7. Dick, J., Pillichshammer, F.: Strong tractability of multivariate integration of arbitrary high order using digitally shifted polynomial lattice rules. J. Complex. **23**, 436–453 (2007)
8. Dick, J., Pillichshammer, F.: Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration. Cambridge University Press, Cambridge, U.K. (2010)
9. Dick, J., Sloan, I.H., Wang, X., Woźniakowski, H.: Good lattice rules in weighted Korobov spaces with general weights. Numerische Mathematik **103**, 63–97 (2006)
10. Efron, B., Stein, C.: The jackknife estimator of variance. Ann. Stat. **9**, 586–596 (1981)
11. Genz, A.: Testing multidimensional integration routines. In: Proceedings of the International Conference on Tools, Methods and Languages for Scientific and Engineering Computation, pp. 81–94. Elsevier North-Holland (1984)
12. Gilbert, A., Kuo, F., Sloan, I.: Hiding the weights-CBC black box algorithms with a guaranteed error bound. Math. Comput. Simul. **143**, 202–214 (2018)
13. Goda, T.: Good interlaced polynomial lattice rules for numerical integration in weighted Walsh spaces. J. Comput. Appl. Math. **285**, 279–294 (2015)
14. Goda, T., Dick, J.: Construction of interlaced scrambled polynomial lattice rules of arbitrary high order. Found. Comput. Math. **15**, 1245–1278 (2019)
15. Goda, T., Suzuki, K.: Recent Advances in Higher Order Quasi-Monte Carlo Methods, pp. 69–102. De Gruyter (2019)
16. Goda, T., Suzuki, K., Yoshiki, T.: An explicit construction of optimal order Quasi-Monte Carlo rules for smooth integrands. SIAM J. Numer. Anal. **54**, 2664–2683 (2016)
17. Goda, T., Suzuki, K., Yoshiki, T.: Optimal order Quasi-Monte Carlo integration in weighted Sobolev spaces of arbitrary smoothness. SIAM J. Numer. Anal. **37**, 505–518 (2017)
18. Goda, T., Suzuki, K., Yoshiki, T.: Optimal order quadrature error bounds for infinite-dimensional higher-order digital sequences. Found. Comput. Math. **18**, 433–458 (2018)
19. Hickernell, F.J.: A generalized discrepancy and quadrature error bound. Math. Comput. **67**(221), 299–322 (1998)
20. Hickernell, F.J.: Lattice rules: How well do they measure up? In: Hellekalek, P., Larcher, G. (eds.) Random and Quasi-Random Point Sets. Lecture Notes in Statistics, vol. 138, pp. 109–166. Springer, New York (1998)
21. Hickernell, F.J.: Obtaining $O(N^{-2+\epsilon})$ convergence for lattice quadrature rules. In: Fang, K.T., Hickernell, F.J., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 274–289. Springer, Berlin (2002)
22. Hickernell, F.J., Hong, H.S., L'Ecuyer, P., Lemieux, C.: Extensible lattice sequences for Quasi-Monte Carlo quadrature. SIAM J. Sci. Comput. **22**(3), 1117–1138 (2001)
23. Hickernell, F.J., Yue, R.X.: The mean square discrepancy of scrambled (t, s)-sequences. SIAM J. Numer. Anal. textbf38(4), 1089–1112 (2001)
24. Hong, H.S., Hickernell, F.H.: Algorithm 823: implementing scrambled digital sequences. ACM Trans. Math. Softw. **29**, 95–109 (2003)
25. Joe, S., Kuo, F.Y.: Constructing Sobol sequences with better two-dimensional projections. SIAM J. Sci. Comput. **30**(5), 2635–2654 (2008)
26. L'Ecuyer, P.: Polynomial integration lattices. In: Niederreiter, H. (ed.) Monte Carlo and Quasi-Monte Carlo Methods 2002, pp. 73–98. Springer, Berlin (2004)
27. L'Ecuyer, P.: Quasi-Monte Carlo methods with applications in finance. Financ. Stoch. **13**(3), 307–349 (2009)

28. L'Ecuyer, P.: SSJ: Stochastic Simulation in Java (2016). http://simul.iro.umontreal.ca/ssj/. Accessed 9 Aug 2021
29. L'Ecuyer, P., Buist, E.: Simulation in Java with SSJ. In: Proceedings of the 2005 Winter Simulation Conference, pp. 611–620. IEEE Press, Piscataway, NJ (2005)
30. L'Ecuyer, P., Lemieux, C.: Variance reduction via lattice rules. Manag. Sci. **46**(9), 1214–1235 (2000)
31. L'Ecuyer, P., Lemieux, C.: Recent advances in randomized Quasi-Monte Carlo methods. In: Dror, M., L'Ecuyer, P., Szidarovszky, F. (eds.) Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications, pp. 419–474. Kluwer Academic, Boston (2002)
32. L'Ecuyer, P., Munger, D.: Constructing adapted lattice rules using problem-dependent criteria. In: Proceedings of the 2012 Winter Simulation Conference, pp. 373–384. IEEE Press (2012)
33. L'Ecuyer, P., Munger, D.: On figures of merit for randomly-shifted lattice rules. In: Woźniakowski, H., Plaskota, L. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2010, pp. 133–159. Springer, Berlin (2012)
34. L'Ecuyer, P., Munger, D.: Algorithm 958: Lattice builder: a general software tool for constructing rank-1 lattice rules. ACM Trans. Math. Softw. **42**(2), Article 15 (2016)
35. Lemieux, C., Cieslak, M., Luttmer, K.: RandQMC User's Guide: A Package for Randomized Quasi-Monte Carlo Methods in C (2004). Software user's guide, available at http://www.math.uwaterloo.ca/~clemieux/randqmc.html
36. Lemieux, C., L'Ecuyer, P.: Randomized polynomial lattice rules for multivariate integration and simulation. SIAM J. Sci. Comput. **24**(5), 1768–1789 (2003)
37. Marion, P., Godin, M., L'Ecuyer, P.: An algorithm to compute the t-value of a digital net and of its projections. J. Comput. Appl. Math. **371**(June), 112669 (2020)
38. Matoušek, J.: On the L_2-discrepancy for anchored boxes. J. Complex. **14**, 527–556 (1998)
39. Niederreiter, H.: Low-discrepancy point sets obtained by digital constructions over finite fields. Czechoslovak Math. J. **42**, 143–166 (1992)
40. Niederreiter, H.: Random number generation and Quasi-Monte Carlo methods. In: SIAM CBMS-NSF Regional Conference Series in Mathematics, vol. 63. SIAM (1992)
41. Nuyens, D.: Fast component-by-component constructions (2012). http://people.cs.kuleuven.be/~dirk.nuyens/fast-cbc/, http://people.cs.kuleuven.be/~dirk.nuyens/fast-cbc/
42. Nuyens, D.: The construction of good lattice rules and polynomial lattice rules. In: Kritzer, P., Niederreiter, H., Pillichshammer, F., Winterhof, A. (eds.) Uniform Distribution and Quasi-Monte Carlo Methods: Discrepancy, Integration and Applications, pp. 223–255. De Gruyter (2014)
43. Nuyens, D.: The Magic Point Shop (2020). https://people.cs.kuleuven.be/~dirk.nuyens/qmc-generators/
44. Nuyens, D., Cools, R.: Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces. Math. Comput. **75**, 903–920 (2006)
45. Owen, A.B.: Monte Carlo variance of scrambled equidistribution quadrature. SIAM J. Numer. Anal. **34**(5), 1884–1910 (1997)
46. Owen, A.B.: Scrambled net variance for integrals of smooth functions. Ann. Stat. **25**(4), 1541–1562 (1997)
47. Owen, A.B.: Latin supercube sampling for very high-dimensional simulations. ACM Trans. Model. Comput. Simul. **8**(1), 71–102 (1998)
48. Owen, A.B.: Variance with alternative scramblings of digital nets. ACM Trans. Model. Comput. Simul. **13**(4), 363–378 (2003)
49. Sinescu, V., L'Ecuyer, P.: Variance bounds and existence results for randomly shifted lattice rules. J. Comput. Appl. Math. **236**, 3296–3307 (2012)
50. Sloan, I.H., Joe, S.: Lattice Methods for Multiple Integration. Clarendon Press, Oxford (1994)
51. Sloan, I.H., Rezstov, A.: Component-by-component construction of good lattice rules. Math. Comput. **71**, 262–273 (2002)

52. Sobol', I.M.: The distribution of points in a cube and the approximate evaluation of integrals. U.S.S.R. Comput. Math. and Math. Phys. **7**(4), 86–112 (1967)
53. Sobol, I.M., Asotsky, D.I.: One more experiment on estimating high-dimensional integrals by quasi-Monte Carlo methods. Math. Comput. Simul. **62**(3–6), 255–263 (2003)
54. Yue, R.X., Hickernell, F.J.: The discrepancy and gain coefficients of scrambled digital nets. J. Complex. **18**(1), 135–151 (2002). http://www.sciencedirect.com/science/article/pii/S0885064X01906302

On Dropping the First Sobol' Point

Art B. Owen

Abstract Quasi-Monte Carlo (QMC) points are a substitute for plain Monte Carlo (MC) points that greatly improve integration accuracy under mild assumptions on the problem. Because QMC can give errors that are $o(1/n)$ as $n \to \infty$, and randomized versions can attain root mean squared errors that are $o(1/n)$, changing even one point can change the estimate by an amount much larger than the error would have been and worsen the convergence rate. As a result, certain practices that fit quite naturally and intuitively with MC points can be very detrimental to QMC performance. These include thinning, burn-in, and taking sample sizes such as powers of 10, when the QMC points were designed for different sample sizes. This article looks at the effects of a common practice in which one skips the first point of a Sobol' sequence. The retained points ordinarily fail to be a digital net and when scrambling is applied, skipping over the first point can increase the numerical error by a factor proportional to $\sqrt{n}$ where n is the number of function evaluations used.

Keywords Burn-in · Scrambled nets · Thinning

1 Introduction

A Sobol' sequence is an infinite sequence of points $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots \in [0, 1]^d$ constructed to fill out the unit cube with low discrepancy, meaning that a measure of the distance between the discrete uniform distribution on $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ and the continuous uniform distribution on $[0, 1]^d$ is made small. These points are ordinarily used to approximate

$$\mu = \int_{[0,1]^d} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \quad \text{by} \quad \hat{\mu} = \hat{\mu}_{\boldsymbol{u},1} = \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{u}_i).$$

A. B. Owen (✉)
Stanford University, Stanford, CA 94305, USA
e-mail: owen@stanford.edu

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_4

The reason for calling this estimate $\hat{\mu}_{\boldsymbol{u},1}$ will become apparent later. Sobol' sequences are often used to estimate expectations with respect to unbounded random variables, such as Gaussians. In such cases f subsumes a transformation from the uniform distribution on $[0,1]^d$ to some other more appropriate distribution. This article uses 1-based indexing, so that the initial point is $\boldsymbol{u}_1$. Sometimes 0-based indexing is used, and then the initial point is denoted $\boldsymbol{u}_0$. Both indexing conventions are widespread in mathematics and software for Sobol' points and both have their benefits. Whichever convention is used, the first point should not be dropped.

The initial point of the Sobol' sequence is $\boldsymbol{u}_1 = (0, 0, \ldots, 0)$. A common practice is to skip that point, similar to the burn-in practice in Markov chain Monte Carlo (MCMC). One then estimates μ by

$$\hat{\mu} = \hat{\mu}_{\boldsymbol{u},2} = \frac{1}{n}\sum_{i=2}^{n+1} f(\boldsymbol{u}_i).$$

One reason to skip the initial point is that a transformation to a Gaussian distribution might make the initial Gaussian point infinite. That is problematic not just for integration problems but also when f is to be evaluated at the design points to create surrogate models for Bayesian optimization [1, 13]. If one skips the initial point, then the next point in a Sobol' sequence is usually $(1/2, 1/2, \ldots, 1/2)$. While that is an intuitively much more reasonable place to start, starting there has detrimental consequences and there are better remedies, described here.

A discussion about whether to drop the initial point came up in the plenary tutorial of Fred Hickernell at MCQMC 2020 about QMCPy [5] software for QMC, discussed in [6]. The issue has been discussed by the pytorch [39] community at https://github.com/pytorch/pytorch/issues/32047, and the scipy [47] community at https://github.com/scipy/scipy/pull/10844, which are both incorporating QMC methods. QMC and RQMC code for scipy is documented at https://scipy.github.io/devdocs/reference/stats.qmc.html.

This article shows that skipping even one point of the Sobol' sequence can be very detrimental. The resulting points are no longer a digital net in general, and in the case of scrambled Sobol' points, skipping a point can bring about an inferior rate of convergence, making the estimate less accurate by a factor that is roughly proportional to $\sqrt{n}$.

A second difficulty with Sobol' sequence points is that it is difficult to estimate the size $|\hat{\mu} - \mu|$ of the integration error from the data. The well-known Koksma-Hlawka inequality [15] bounds $|\hat{\mu} - \mu|$ by the product of two unknown quantities that are extremely hard to compute, and while tight for some worst case integrands, it can yield an extreme overestimate of the error, growing ever more conservative as the dimension d increases.

Randomly scrambling the Sobol' sequence points preserves their balance properties and provides a basis for uncertainty quantification. Scrambling turns points $\boldsymbol{u}_i$ into random points $\boldsymbol{x}_i \sim \mathsf{U}[0,1]^d$. The points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are not independent. Instead they retain the digital net property of Sobol' points and consequent accu-

racy properties. The result is randomized QMC (RQMC) points. RQMC points also have some additional accuracy properties stemming from the randomization. With scrambled Sobol' points, we estimate μ by

$$\hat{\mu} = \hat{\mu}_{\boldsymbol{x},1} = \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i).$$

One can estimate the mean squared error using R independent replicates of the n-point RQMC estimate $\hat{\mu}_{\boldsymbol{x},1}$. It is also possible to drop the first point in RQMC, estimating μ by

$$\hat{\mu} = \hat{\mu}_{\boldsymbol{x},2} = \frac{1}{n}\sum_{i=2}^{n+1} f(\boldsymbol{x}_i).$$

The purpose of this paper is to show that $\hat{\mu}_{\boldsymbol{x},1}$ is a much better choice than $\hat{\mu}_{\boldsymbol{x},2}$.

Many implementations of a Sobol' sequence will produce $n = 2^m$ points $\boldsymbol{u}_i \in \{0, 1/n, 2/n, \ldots, (n-1)/n\}^d \subset [0,1)^d$. In that case, there is a safer way to avoid having a point at the origin than skipping the first point. We can use $\boldsymbol{u}_i + 1/(2n)$ componentwise and still have a digital net. This is reasonable if we have already decided on the value of n to use. It does not work to add that same value $1/(2n)$ to the next 2^m points and subsequent values. For one thing, the result may produce values on the upper boundary of $[0,1]^d$ in the very next batch and will eventually place points outside of $[0,1]^d$. It remains better to scramble the Sobol' points.

We will judge the accuracy of integration via scrambled Sobol' points through $\mathbb{E}((\hat{\mu}-\mu)^2)^{1/2}$, the root mean squared error (RMSE). Plain Monte Carlo (MC) attains an RMSE of $O(n^{-1/2})$ for integrands $f \in L^2[0,1]^d$.

This paper is organized as follows. Section 2 defines digital nets and shows that skipping over the first point can destroy the digital net property underlying the analysis of Sobol' sequences. It also presents properties of scrambled digital nets. Section 3 shows some empirical investigations on some very simple and favorable integrands where $\hat{\mu}_{\boldsymbol{x},1}$ has an RMSE very near to the rate $O(n^{-3/2})$ while $\hat{\mu}_{\boldsymbol{x},2}$ has an RMSE very near to $O(n^{-1})$. These are both in line with what we expect from asymptotic theory. The relevance is not that our integrands are as trivial as those examples, but rather that when realistic integrands are well approximated by such simple ones we get accuracy comparable to using those simple functions as control variates [16] but without us having to search for control variates. In skipping the first point we stand to lose a lot of accuracy in integrating the simple functions and others close to them. There is also no theoretical reason to expect $\hat{\mu}_{\boldsymbol{x},2}$ to have a smaller RMSE than $\hat{\mu}_{\boldsymbol{x},1}$ does, and so there is a Pascal's wager argument against dropping the first point. Section 4 looks at a ten dimensional function representing the weight of an airplane wing as a function of the way it was made. We see there that skipping the first point is very detrimental. Section 5 considers some very special cases where

burn-in might be harmless, recommends against using round number sample sizes and thinning for QMC points, and discusses distributing QMC points over multiple parallel processors.

2 Digital Nets and Scrambling

In this section we review digital nets and describe properties of their scrambled versions. The points from Sobol' sequences provide the most widely used example of digital nets. For details of their construction and analysis, see the monographs [11, 28]. There are numerous implementations of Sobol' sequences [2, 18, 45]. They differ in what are called 'direction numbers' and they can also vary in the order with which the points are generated. The numerical results here use direction numbers from [18] via an implementation from Nuyens' magic point shop, described in [22] and scrambled as in [29]. The Sobol' and scrambled Sobol' points in this paper were generated using the R function `rsobol` that appears in http://statweb.stanford.edu/~owen/code/ along with some documentation. That code also includes the faster and more space efficient scrambling of Matousek [26].

We begin with the notion of elementary intervals, which are special hyper-rectangular subsets of $[0, 1)^d$. For an integer base $b \geqslant 2$, a dimension $d \geqslant 1$, a vector $\boldsymbol{k} = (k_1, \ldots, k_d)$ of integers $k_j \geqslant 0$ and a vector $\boldsymbol{c} = (c_1, \ldots, c_d)$ of integers with $0 \leqslant c_j < b^{k_j}$, the Cartesian product

$$E(\boldsymbol{k}, \boldsymbol{c}) = \prod_{j=1}^{d} \left[\frac{c_j}{b^{k_j}}, \frac{c_j + 1}{b^{k_j}} \right)$$

is an elementary interval in base b. It has volume $b^{-|\boldsymbol{k}|}$ where $|\boldsymbol{k}| = \sum_{j=1}^{d} k_j$.

Speaking informally, the set $E(\boldsymbol{k}, \boldsymbol{c})$ has a proportion $b^{-|\boldsymbol{k}|}$ of the volume of $[0, 1]^d$ and so it 'deserves' to get (i.e., contain) $nb^{-|\boldsymbol{k}|}$ points when we place n points inside $[0, 1]^d$. Digital nets satisfy that condition for certain $\boldsymbol{k}$. We use the following definitions from Niederreitter [27].

Definition 1 For integers $m \geqslant t \geqslant 0$, the $n = b^m$ points $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in [0, 1]^d$ are a (t, m, d)-net in base $b \geqslant 2$, if every elementary interval $E(\boldsymbol{k}, \boldsymbol{c}) \subset [0, 1]^d$ of volume b^{t-m} contains exactly b^t of the points $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$.

Every elementary interval that 'deserves' b^t points of the digital net, gets that many of them. When we speak of digital nets we ordinarily mean (t, m, d)-nets though some authors reserve the term 'digital' to refer to specific construction algorithms rather than just the property in Definition 1.

Definition 2 For integers $t \geqslant 0, b \geqslant 2$ and $d \geqslant 1$, the infinite sequence $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots \in [0, 1]^d$ is a (t, d)-sequence in base b if $\boldsymbol{u}_{(r-1)b^m+1}, \ldots, \boldsymbol{u}_{rb^m}$ is a (t, m, d)-net in base b for any integers $m \geqslant t$ and $r \geqslant 1$.

Sobol' sequences [42] are (t, d)-sequences in base 2. From Definition 2, we see that the first 2^m points of a Sobol' sequence are a (t, m, d)-net in base 2 for any $m \geqslant t$. So are the second 2^m points, and if we merge both of those point sets, we get a $(t, m+1, d)$-net in base 2. We can merge the first two of those to get a $(t, m+2, d)$-net in base 2 and so on ad infinitum.

Given b, m and d, smaller values of t are better. It is not always possible to have $t = 0$ and the best possible t increases monotonically with d. The best known values of t for (t, d)-sequences and (t, m, d)-nets are given in the online MinT web site [40]. The published t value for a Sobol' sequence might be conservative in that the first b^m points of the Sobol' sequence can possibly be a (t', m, d)-net for some $t' < t$.

The proven properties of digital nets including those taken from Sobol' sequences derive from their balanced sampling of elementary intervals. The analysis path can be via discrepancy [28] or Haar wavelets [43] or Walsh functions [11].

The left panel in Fig. 1 shows the first 16 points of a Sobol' sequence in two dimensions. Fifteen of them are small solid disks and one other is represented by concentric circles at the origin. Those points form a $(0, 4, 2)$-net in base 2. Reference lines divide the unit square into a 4×4 grid of elementary intervals of size $1/4 \times 1/4$. Each of those has one of the 16 points, often at the lower left corner. Recall that elementary intervals include their lower boundary but not their upper boundary. Finer reference lines partition the unit square into 16 strips of size $1 \times 1/16$. Each of those has exactly one point of the digital net. The same holds for the 16 rectangles of each of these shapes: $1/2 \times 1/8$, $1/8 \times 1/2$ and $1/16 \times 1$. All told, those 16 points have balanced 80 elementary intervals and the number of balanced intervals grows rapidly with m and d.

The point $\boldsymbol{u}_1 = (0, 0)$ is problematic as described above. If we skip it and take points $\boldsymbol{u}_2, \ldots, \boldsymbol{u}_{17}$ then we replace it with the large solid disk at $(1/32, 17/32)$. Doing that leaves the lower left $1/4 \times 1/4$ square empty and puts two points into a square above it. The resulting 16 points now fail to be a $(0, 4, 2)$-net.

The introduction mentioned some randomizations of digital nets. There is a survey of RQMC in [24]. For definiteness, we consider the nested uniform scramble from [29]. Applying a nested uniform scramble to a (t, d)-sequence $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots$ in base b yields points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ that individually satisfy $\boldsymbol{x}_i \sim \mathsf{U}[0, 1]^d$ and collectively are a (t, d)-net in base b with probability one. The estimate $\hat{\mu}_{\boldsymbol{x},1}$ then satisfies $\mathbb{E}(\hat{\mu}_{\boldsymbol{x},1}) = \mu$ by uniformity of $\boldsymbol{x}_i$. The next paragraphs summarize some additional properties of scrambled nets.

If $f \in L^{1+\epsilon}[0, 1]^d$ for some $\epsilon > 0$ then [37] show that $\Pr(\lim_{m\to\infty} \hat{\mu}_{\boldsymbol{x},1} = \mu) = 1$, where the limit is through (t, m, d)-nets formed by initial b^m subsequences the (t, d)-sequence of $\boldsymbol{x}_i$. If $f \in L^2[0, 1]^d$ then $\mathrm{var}(\hat{\mu}_{\boldsymbol{x},1}) = o(1/n)$ as $n = b^m \to \infty$ [30]. That is, the RMSE is $o(n^{-1/2})$, superior to MC. Evidence of convergence rates for RQMC better than $n^{-1/2}$ have been seen for some unbounded integrands from financial problems. For instance variance reduction factors with respect to MC have been seen to increase with sample size in [23].

The usual regularity condition for plain MC is that $f(\boldsymbol{x})$ has finite variance σ^2 and the resulting RMSE is $\sigma n^{-1/2}$. When $f \in L^2[0, 1]^d$ with variance σ^2 then scrambled net sampling with $n = b^m$ satisfies

$$\mathrm{RMSE}(\hat{\mu}_{\boldsymbol{x},1}) \leqslant \Gamma^{1/2}\sigma n^{-1/2} \tag{1}$$

for some $\Gamma < \infty$ [32]. For digital nets in base 2, such as those of Sobol', it is known that Γ is a power of two no larger than 2^{t+d-1} [38]. Equation (1) describes a worst case $f \in L^2[0,1]^d$ that maximizes the ratio of the RMSE for RQMC to that of MC.

The accuracy of QMC points is most commonly described by a worst case analysis with $|\mu - \hat{\mu}| = O(n^{-1}\log(n)^{d-1})$ when f has bounded variation in the sense of Hardy and Krause (BVHK). These powers of $\log(n)$ are not negligible for practically relevant values of n, when d is moderately large. Then the bound gives a misleadingly pessimistic idea of the accuracy one can expect. The bound in (1) shows that the RMSE of scrambled nets is at most $\Gamma^{1/2}\sigma/\sqrt{n}$, a bound with no powers of $\log(n)$. This holds for $f \in L^2$, which then includes any f in BVHK as well as many others of practical interest, such as some unbounded integrands. Note that integrands in BVHK must be bounded and they are also Riemann integrable [37], and so they are in L^2.

Under further smoothness conditions on f, $\mathrm{RMSE}(\hat{\mu}_{\boldsymbol{x},1}) = O(n^{-3/2}(\log n)^{(d-1)/2})$. This was first noticed in [31] with a correction in [33]. The weakest known sufficient conditions are a generalized Lipschitz condition from [48]. The condition in [33] is that for any nonempty $u \subseteq \{1,\dots,d\}$ the mixed partial derivative of f taken once with respect to each index $j \in u$ is continuous on $[0,1]^d$. To reconcile the appearance and non-appearance of logarithmic factors, those two results give $\mathrm{RMSE}(\hat{\mu}_{\boldsymbol{x},1}) \leqslant \min(\Gamma^{1/2}\sigma n^{-1/2}, A_n)$ for some sequence $A_n = O(n^{-3/2}\log(n)^{(d-1)/2})$. The logarithmic factor can degrade the $n^{-3/2}$ rate but only subject to a cap on performance relative to plain MC. Finally, Loh [25] proves a central limit theorem for $\hat{\mu}_{\boldsymbol{x},1}$ when $t = 0$.

The right panel of Fig. 1 shows a nested uniform scramble of the points in the left panel. The problematic point $\boldsymbol{u}_1$ becomes a uniformly distributed point in the square, and is no longer on the boundary. If we replace it by $\boldsymbol{u}_{17}$ then just as in the unscrambled case, there is an empty $1/4 \times 1/4$ elementary interval, and another one with two points.

There is a disadvantage to $\hat{\mu}_{\boldsymbol{x},2}$ compared to $\hat{\mu}_{\boldsymbol{x},1}$ when the latter attains a root mean squared error $O(n^{-3/2+\epsilon})$, for then

$$\hat{\mu}_{\boldsymbol{x},2} = \hat{\mu}_{\boldsymbol{x},1} + \frac{1}{n}\big(f(\boldsymbol{x}_{n+1}) - f(\boldsymbol{x}_1)\big). \tag{2}$$

The term $(f(\boldsymbol{x}_{n+1}) - f(\boldsymbol{x}_1))/n = O(1/n)$ will ordinarily decay more slowly than $|\hat{\mu}_{\boldsymbol{x},1} - \mu|$. Then skipping the first point will actually make the rate of convergence worse. A similar problem happens if one simply ignores $\boldsymbol{x}_1$ and averages the $n-1$ points $f(\boldsymbol{x}_2)$ through $f(\boldsymbol{x}_n)$. A related issue is that when equally weighted integration rules have errors $O(n^{-r})$ for $r > 1$, this rate can only realistically take place at geometrically separated values of n. See [34, 44]. The higher order digital nets of [9] attain $o(1/n)$ errors under suitable regularity conditions and their randomizations in [10] attain RMSEs of $o(1/n)$. The argument against skipping the first point also applies to these methods.

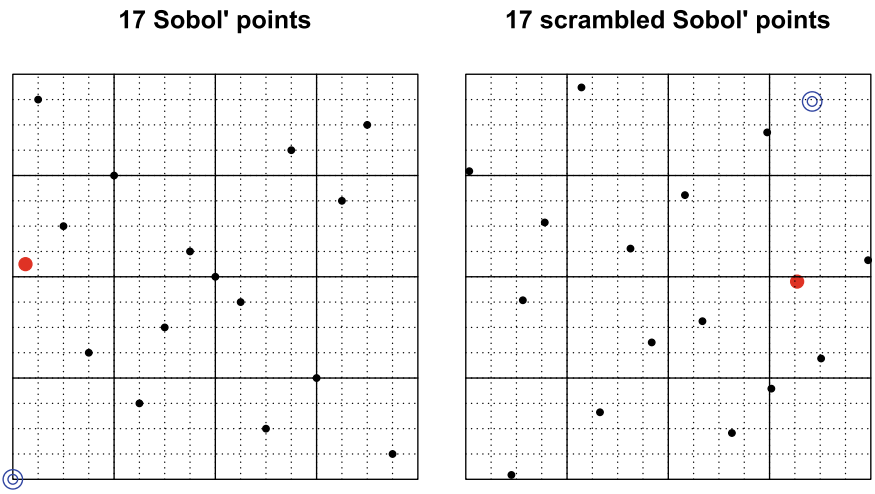


Fig. 1 The left panel shows the first 17 Sobol' points in $[0, 1]^2$. The initial point at (0, 0) is shown in concentric circles. The 17'th point is shown as a large disk. Solid reference lines partition $[0, 1]^2$ into 16 congruent squares. Dashed reference lines partition it into 256 congruent squares. The right panel shows a nested uniform scramble of these 17 points

3 Synthetic Examples

Here we look at some very simple modest dimensional integrands. They fit into a 'best case' case analysis for integration, motivated as follows. We suppose that some sort of function $g(\boldsymbol{x})$ is extremely favorable for a method and also that it resembles the actual integrand. We may write

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) + \varepsilon(\boldsymbol{x}).$$

In the favorable cases, ε is small and g is easily integrated. For classical quadratures g may be a polynomial [8]. For digital nets, some functions g may have rapidly converging Walsh series [11], others are sums of functions of only a few variables at a time [3]. For lattice rules [41], a favorable g has a rapidly converging Fourier series. The favorable cases work well because

$$\frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i) = \frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{x}_i) + \frac{1}{n}\sum_{i=1}^{n} \varepsilon(\boldsymbol{x}_i)$$

with the first term having small error because it is well suited to the method and the second term having small error because $\varepsilon(\cdot)$ has a small norm and we take an equal weight sample of it instead of using large weights of opposite signs. A good match between method and g saves us the chore of searching for one or more control variates. Choosing cases where a method ought to work is like the positive controls

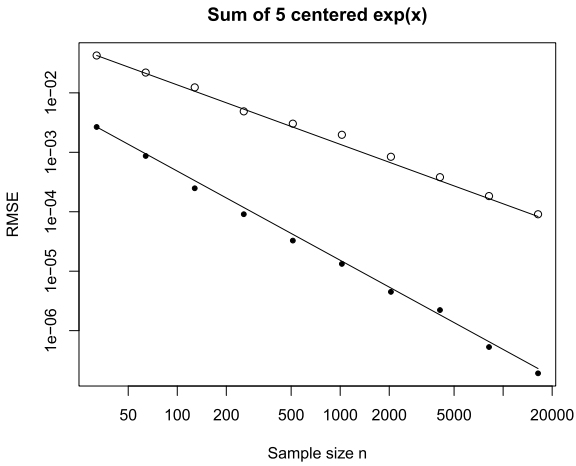


Fig. 2 Solid points show RMSE for scrambled Sobol' estimate $\hat{\mu}_{\boldsymbol{x},1}$ versus n from $R = 10$ replicates. A reference line parallel to $n^{-3/2}$ goes through the first solid point. Open points show RMSE for scrambled Sobol' estimates $\hat{\mu}_{\boldsymbol{x},2}$ which drop the initial zero. A reference line parallel to n^{-1} goes through the first open point

used in experimental science. We can use them to verify that the method or its numerical implementation work as expected on the cases they were designed for. There can and will be unfavorable cases in practice. Measuring the sample variance under replication provides a way to detect that.

Here we consider some cases where scrambled nets should work well. The first is

$$g_0(\boldsymbol{x}) = \sum_{j=1}^{d} \bigl(e^{x_j} - e + 1\bigr), \tag{3}$$

which clearly has $\mu = 0$. This sum of centered exponentials is smooth and additive. It is thus very simple for QMC and RQMC. It is unlikely that anybody turns to RQMC for this function but as remarked above the integrand one has may be close to such a simple function.

Figure 2 shows the RMSE for this function g_0 based on $R = 10$ independent replicates of both $\hat{\mu}_{\boldsymbol{x},1}$ and $\hat{\mu}_{\boldsymbol{x},2}$. Reference lines show a clear pattern. The error follows a reference line parallel to $n^{-3/2}$ on a log-log plot for $\hat{\mu}_{\boldsymbol{x},1}$. For $\hat{\mu}_{\boldsymbol{x},2}$, the reference line is parallel to n^{-1}. These slopes are exactly what we would expect from the underlying theory, the first from [31] and the second from Equation (2). In both cases the line goes through the data for $n = 32$ and is then extrapolated to $n = 2^{14} = 16{,}384$ with the given slopes. That is a more severe test for the asymptotic theory than fitting by least squares would be. In this instance, the asymptotic theory is already close to the measurements by $n = 32$.

An earlier version of this article used $g_0(\boldsymbol{x}) = \sum_{j=1}^{d} x_j$ instead of the function g_0 above. The RMSEs for that function also closely follow the predicted rates. It is not however as good a test case because it is antisymmetric about $\boldsymbol{x} = (1/2, \ldots, 1/2)$, meaning that $(g_0(\boldsymbol{x}) + g_0(\tilde{\boldsymbol{x}}))/2 = \mu$ for all $\boldsymbol{x}$, where $\tilde{\boldsymbol{x}} = 1 - \boldsymbol{x}$ componentwise.

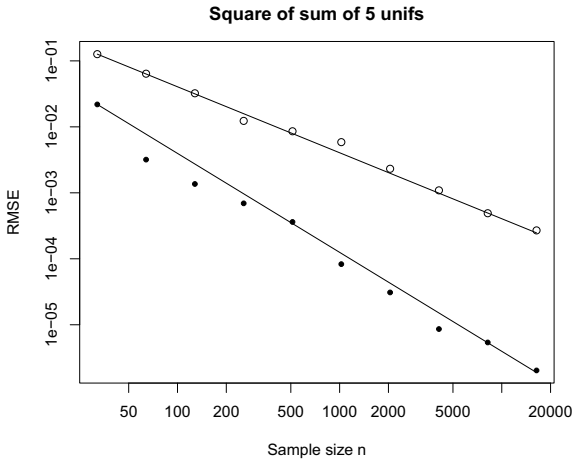


Fig. 3 Solid points show RMSE for scrambled Sobol' estimate $\hat{\mu}_{\boldsymbol{x},1}$ versus n from $R = 10$ replicates. A reference line parallel to $n^{-3/2}$ goes through the first solid point. Open points show RMSE for scrambled Sobol' estimates $\hat{\mu}_{\boldsymbol{x},2}$ which drop the initial zero. A reference line parallel to n^{-1} goes through the first open point

If we use such an antisymmetric function, then we will get highly accurate results just from having a nearly antithetic set of evaluation points that may or may not be equidistributed.

The second function is

$$g_1(\boldsymbol{x}) = \left(\sum_{j=1}^{d} x_j\right)^2. \tag{4}$$

Unlike g_0 this function is not additive. It has interactions of order 2 but no higher in the functional ANOVA decomposition [17, 43] and it also has a substantial additive component. It is not antisymmetric about $(1/2, 1/2, \ldots, 1/2)$. It has $\mu = d/3 + d(d-1)/4$. Figure 3 shows the RMSE for $\hat{\mu}_{\boldsymbol{x},1}$ and $\hat{\mu}_{\boldsymbol{x},2}$. Once again they follow reference lines parallel to $n^{-3/2}$ and n^{-1} respectively. Asymptotic theory predicts a mean squared error with a component proportional to n^{-3} and a second component proportional to $\log(n)n^{-3}$ that would eventually dominate the first, leading to an RMSE that approaches $n^{-3/2}\log(n)^{1/2}$.

Next we look at a product

$$g_2(\boldsymbol{x}) = \prod_{j=1}^{d} (e^{x_j} - e + 1).$$

This function has $\mu = 0$ for any d. It is surprisingly hard for (R)QMC to handle this function for modest d, much less large d. It is dominated by 2^d spikes of opposite signs around the corners of $[0, 1]^d$. It may also be extra hard for Sobol' points compared to alternatives, because Sobol' points often have rectangular blocks that alternate between double the uniform density and emptiness. In a functional ANOVA

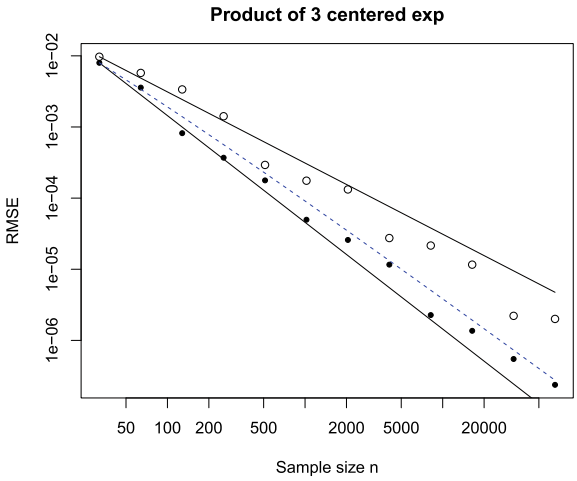


Fig. 4 The integrand is a product of 3 centered exponentials. Solid points show RMSE for scrambled Sobol' estimate $\hat{\mu}_{\boldsymbol{x},1}$ versus n from $R = 10$ replicates. A reference line parallel to $n^{-3/2}$ goes through the first solid point. Open points show RMSE for scrambled Sobol' estimates $\hat{\mu}_{\boldsymbol{x},2}$ which drop the initial zero. A reference line parallel to n^{-1} goes through the first open point. A dashed reference line through the first solid point decays as $\log(n)/n^{3/2}$

decomposition, it is purely d-dimensional in that the only non-zero variance component is the one involving all d variables. Asymptotic theory predicts an RMSE of $O(n^{-3/2}\log(n)^{(d-1)/2})$.

Figure 4 shows results for $d = 3$ and this $g_2(\boldsymbol{x})$. The rate for $\hat{\mu}_{\boldsymbol{x},1}$ shows up as slightly worse than $n^{-3/2}$ while the one for $\hat{\mu}_{\boldsymbol{x},2}$ appears to be slightly better than n^{-1}. Both are much better than $O(n^{-1/2})$. Putting in the predicted logarithmic factor improves the match between asymptotic prediction and empirical outcome for $\hat{\mu}_{\boldsymbol{x},1}$. It is not clear what can explain $\hat{\mu}_{\boldsymbol{x},2}$ doing better here than the asymptotic prediction. Perhaps the asymptotics become descriptive of actual errors at much larger n for this function than for the others. Judging by eye it is possible that the convergence rate is worse when the first point is dropped, but the evidence is not as clear as in the other figures where the computed values so closely follow theoretical predictions. There is an evident benefit to retaining the initial point that at a minimum manifests as a constant factor of improvement.

In some of the above examples the asymptotic theory fit very well by $n = 32$. One should not expect this in general. It is more reasonable to suppose that that is a consequence of the simple form of the integrands studied in this section. For these integrands the strong advantage of retaining the original point shows in both theory and empirical values. There is no countervailing theoretical reason to support dropping the first point.

Table 1 Variables and their ranges for the wing weight function

Variable	Range	Meaning
S_w	[150, 200]	Wing area (ft^2)
W_{fw}	[220, 300]	Weight of fuel in the wing (lb)
A	[6, 10]	Aspect ratio
Λ	[−10, 10]	Quarter-chord sweep (degrees)
q	[16, 45]	Dynamic pressure at cruise (lb/ft^2)
λ	[0.5, 1]	Taper ratio
t_c	[0.08, 0.18]	Aerofoil thickness to chord ratio
N_z	[2.5, 6]	Ultimate load factor
W_{dg}	[1700,2500]	Flight design gross weight (lb)
W_p	[0.025, 0.08]	Paint weight (lb/ft^2)

4 Wing Weight Function

The web site [46] includes a 10 dimensional function that computes the weight of an airplane's wing based on a physical model of the way the wing is manufactured. While one does not ordinarily want to know the average weight of a randomly manufactured wing, this function is interesting in that it has a real physical world origin instead of being completely synthetic. It is easily integrated by several QMC methods [36] and so it is very likely that it equals $g + \varepsilon$ for a favorable g and a small ε.

The wing weight function is

$$0.036 S_w^{0.758} W_{fw}^{0.0035} \left(\frac{A}{\cos^2(\Lambda)}\right)^{0.6} q^{0.006} \lambda^{0.04} \left(\frac{100 t_c}{\cos(\Lambda)}\right)^{-0.3} (N_x W_{dg})^{0.49} + S_w W_p.$$

The definition and uniform ranges of these variables are given in Table 1.

For this function the standard deviation among 10 independent replicates is used instead of the RMSE. The results are in Fig. 5. Once again there is a strong disadvantage to dropping the first Sobol' point. The RMSE when dropping the first point is very nearly $O(n^{-1})$. The RMSE for not dropping the first point is clearly better. The pattern there is not linear on the log-log scale so we cannot confidently conclude what convergence rate best describes it.

5 Discussion

MC and QMC and RQMC points all come as an $n \times d$ matrix of numbers in [0, 1] that we can then pipe through several functions to change the support set and distribution and finally evaluate a desired integrand. Despite that similarity, there are sharp differences in the properties of QMC and RQMC points that affect how we should use them.

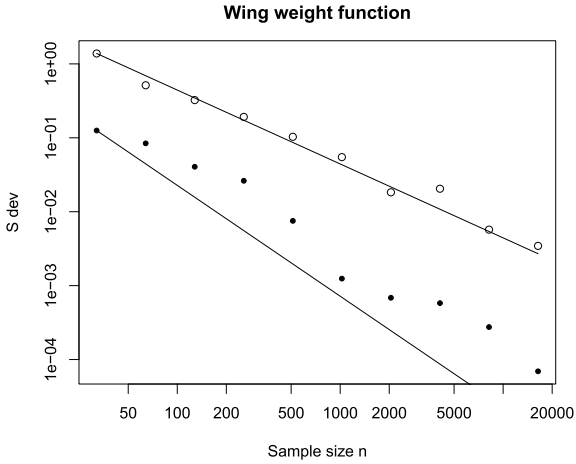


Fig. 5 Solid points show standard deviation for scrambled Sobol' estimate $\hat{\mu}_{\boldsymbol{x},1}$ versus n from $R = 10$ replicates. A reference line parallel to $n^{-3/2}$ goes through the first solid point. Open points show standard deviation for scrambled Sobol' estimates $\hat{\mu}_{\boldsymbol{x},2}$ which drop the initial zero. A reference line parallel to n^{-1} goes through the first open point

This paper has focussed on a small burn-in, dropping just one of the points and picking up the next n. Burn-in makes no difference to plain MC apart from doing some unneeded function evaluations, and it can bring large benefits to MCMC. See the comment by Neal in the discussion [19]. Burn-in typically spoils the digital net property. It is safer to scramble the points which removes the potentially problematic first point at the origin while also increasing accuracy on very favorable functions like those in the examples and also on some unfavorable ones having singularities or other sources of infinite variation in the sense of Hardy and Krause. See [37].

There are some exceptional cases where burn-in of (R)QMC may be harmless. For $d = 1$, any consecutive 2^m points of the van der Corput sequence [7] are a $(0, m, 1)$-net in base 2. As we saw in Fig. 1 that is not always true for $d > 1$. Dropping the first $N = 2^{m'}$ points of a Sobol' sequence for $m' \geq m$ should cause no problems because the next 2^m points are still a (t, m, s)-net. Most current implementations of Sobol' sequences are periodic with $\boldsymbol{x}_i = \boldsymbol{x}_{i+2^M}$ for a value of M that is typically close to 30. Then one could take $m' = M - 1$ allowing one to use m up to $M - 1$.

The Halton sequence [14] has few if any especially good sample sizes n and large burn-ins have been used there. For plain MC points it is natural to use a round number like 1000 or 10^6 of sample points. That can be very damaging in (R)QMC if the points were defined for some other sample size. Using 1000 points of a Sobol' sequence may well be less accurate than using 512. Typical sample sizes are powers of 2 for digital nets and large prime numbers for lattice rules [24, 41]. The Faure sequences [12] use $b = p \geqslant d$ where p is a prime number. With digital nets as with antibiotics, one should take the whole sequence.

Another practice that works well in MCMC, but should not be used in (R)QMC is 'thinning'. In MCMC, thinning can save storage space and in some cases can improve efficiency despite increasing variance [35]. One takes every k'th point, $\boldsymbol{x}_{k\times i}$ for some integer $k > 1$, or in combination with burn-in $\boldsymbol{x}_{B+k\times i}$ for some integer $B \geqslant 1$. To see the problem, consider the very basic van der Corput sequence $x_i \in [0, 1]$. If

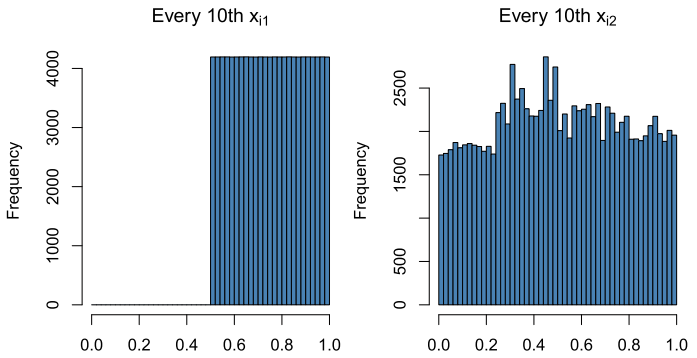


Fig. 6 The left panel shows a histogram of every 10'th $\boldsymbol{x}_{i1}$ from the first 2^{20} Sobol' points. The right panel shows a histogram of every 10'th $\boldsymbol{x}_{i2}$ from the first 2^{20} Sobol' points

$x_i \in [0, 1/2)$ then $x_{i+1} \in [1/2, 1)$. For instance [4] use that observation to point out that simulating a Markov chain with van der Corput points can be problematic. Now suppose that one thins the van der Corput sequence to every second point using $k = 2$. All of the retained points are then in either $[0, 1/2)$ or in $[1/2, 1)$. One will estimate either $2\int_0^{1/2} f(x)\,\mathrm{d}x$ or $2\int_{1/2}^1 f(x)\,\mathrm{d}x$ by using that sequence. The first component of a Sobol' sequence is usually a van der Corput sequence.

Thinning for QMC was earlier considered by [21] who called it 'leaping'. They find interesting results taking every L'th point from a Halton sequence, taking L to be relatively prime to all the bases used in the Halton sequence. Empirically, $L = 409$ was one of the better values. They also saw empirically that leaping in digital nets of Sobol' and Faure lead to non-uniform coverage of the space.

The Matlab R2020a `sobolset` function https://www.mathworks.com/help/stats/sobolset.html as of August 11, 2020 includes a thinning/leaping option through a parameter Leap which is an interval between points, corresponding to $k - 1$ in the discussion above. It also has a parameter Skip, corresponding to burn-in, which is a number of initial points to omit. Fortunately both Leap and Skip are turned off by default. However even having them present is problematic. It is not clear how one should use them safely. The left panel of Fig. 6 shows a histogram of the values $\boldsymbol{x}_{10i,1}$ for $1 \leqslant i \leqslant \lfloor 2^{20}/10 \rfloor$. The right panel shows a histogram of the values $\boldsymbol{x}_{10i,2}$.

Another area where QMC requires more care than plain MC is in parallel computing where a task is to be shared over many processors. When there are p processors working together, one strategy from [20] is to use a $d + 1$ dimensional QMC construction of which one dimension is used to assign input points to processors. Processor $k \in \{0, 1, \ldots, p - 1\}$ gets all the points $\boldsymbol{u}_i$ with $\lfloor p x_{i,c} \rfloor = k$ for some $c \in \{1, 2, \ldots, d + 1\}$. It then uses the remaining d components of $\boldsymbol{u}_i$ in its computation. With this strategy each processor gets a low discrepancy sequence individually which is better than thinning to every p'th point would be. They collectively have a complete QMC point set. See [20] for this and for more references about parallelizing QMC.

Acknowledgements This work was supported by the NSF under grant IIS-1837931, and a grant from Hitachi, Ltd. I thank Fred Hickernell, Pierre L'Ecuyer, Alex Keller, Max Balandat, Michael McCourt, Pamphile Roy and Sergei Kucherenko for stimulating discussions. I think Sifan Liu for catching an error in some code. Thanks to Mike Giles, Arnaud Doucet, Alex Keller and the whole team at ICMS for making MCQMC 2020 happen despite all the pandemic disruption. This paper benefited from comments of two anonymous reviewers. I thank Alex Keller for handling the proceedings volume.

References

1. Balandat, M., Karrer, B., Jiang, D.R., Daulton, S., Letham, B., Wilson, A.G., Bakshy, E.: BoTorch: Bayesian optimization in PyTorch. Technical report (2019). arXiv:1910.06403, Facebook Research
2. Bratley, P., Fox, B.L.: Algorithm 659: implementing Sobol's quasirandom sequence generator. ACM Trans. Math. Softw. **14**(1), 88–100 (1988)
3. Caflisch, R.E., Morokoff, W., Owen, A.B.: Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension. J. Comput. Financ. **1**(1), 27–46 (1997)
4. Caflisch, R.E., Moskowitz, B.: Modified Monte Carlo methods using quasi-random sequences. In: Niederreiter, H., Shiue, P.J.S. (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, pp. 1–16. Springer, New York (1995)
5. Choi, S.C.T., Hickernell, F.J., Jagadeeswaran, R., McCourt, M.J., Sorokin, A.G.: QMCPy: A Quasi-Monte Carlo Python Library (2020). https://doi.org/10.5281/zenodo.3964489, https://qmcsoftware.github.io/QMCSoftware/
6. Choi, S.C.T., Hickernell, F.J., Jagadeeswaran, R., McCourt, M.J., Sorokin, A.G.: Quasi-Monte Carlo software. In: Keller, A. (ed.) Monte Carlo and Quasi-Monte Carlo Methods, MCQMC 2020, Springer Proceedings in Mathematics & Statistics. Springer (2022)
7. van der Corput, J.G.: Verteilungsfunktionen I. Nederl. Akad. Wetensch. Proc. **38**, 813–821 (1935)
8. Davis, P.J., Rabinowitz, P.: Methods of Numerical Integration, 2nd edn. Academic Press, San Diego (1984)
9. Dick, J.: Walsh spaces containing smooth functions and Quasi-Monte Carlo rules of arbitrarily high order. SIAM J. Numer. Anal. **46**(3), 1519–1553 (2008)
10. Dick, J.: Higher order scrambled digital nets achieve the optimal rate of the root mean square error for smooth integrands. Ann. Stat. **39**(3), 1372–1398 (2011)
11. Dick, J., Pillichshammer, F.: Digital Sequences, Discrepancy and Quasi-Monte Carlo Integration. Cambridge University Press, Cambridge (2010)
12. Faure, H.: Discrépance de suites associées à un système de numération (en dimension s). Acta Arithmetica **41**, 337–351 (1982)
13. Frazier, P.I.: A tutorial on Bayesian optimization. Technical report (2018). arXiv:1807.02811
14. Halton, J.H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik **2**(1), 84–90 (1960)
15. Hickernell, F.J.: Koksma-Hlawka Inequality. Statistics Reference Online, Wiley StatsRef (2014)
16. Hickernell, F.J., Lemieux, C., Owen, A.B.: Control variates for Quasi-Monte Carlo (with discussion). Stat. Sci. **20**(1), 1–31 (2005)
17. Hoeffding, W.: A class of statistics with asymptotically normal distribution. Ann. Math. Stat. **19**(3), 293–325 (1948)
18. Joe, S., Kuo, F.Y.: Constructing Sobol' sequences with better two-dimensional projections. SIAM J. Sci. Comput. **30**(5), 2635–2654 (2008)
19. Kass, R.E., Carlin, B.P., Gelman, A., Neal, R.M.: Markov chain Monte Carlo in practice: a roundtable discussion. Am. Stat. **52**(2), 93–100 (1998)

20. Keller, A., Grünschloß, L.: Parallel Quasi-Monte Carlo integration by partitioning low discrepancy sequences. In: Plaskota, L., Woźniakowski, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2010, pp. 487–498. Springer (2012). http://gruenschloss.org/parqmc/parqmc.pdf
21. Kocis, L., Whiten, W.J.: Computational investigations of low-discrepancy sequences. ACM Trans. Math. Softw. **23**(2), 266–294 (1997). https://doi.org/10.1145/264029.264064
22. Kuo, F.Y., Nuyens, D.: Application of Quasi-Monte Carlo methods to elliptic PDEs with random diffusion coefficients: a survey of analysis and implementation. Found. Comput. Math. **16**(6), 1631–1696 (2016)
23. L'Ecuyer, P.: Quasi-Monte Carlo methods with applications in finance. Financ. Stoch. **13**(3), 307–349 (2009)
24. L'Ecuyer, P., Lemieux, C.: Variance reduction via lattice rules. Manag. Sci. **46**(9), 1214–1235 (2000)
25. Loh, W.L.: On the asymptotic distribution of scrambled net quadrature. Ann. Stat. **31**(4), 1282–1324 (2003)
26. Matoušek, J.: On the L^2-discrepancy for anchored boxes. J. Complex. **14**(4), 527–556 (1998)
27. Niederreiter, H.: Point sets and sequences with small discrepancy. Monatshefte für Mathematik **104**(4), 273–337 (1987)
28. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. SIAM, Philadelphia, PA (1992)
29. Owen, A.B.: Randomly permuted (t, m, s)-nets and (t, s)-sequences. In: Niederreiter, H., Shiue, P.J.S. (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, pp. 299–317. Springer, New York (1995)
30. Owen, A.B.: Monte Carlo variance of scrambled net quadrature. SIAM J. Numer. Anal. **34**(5), 1884–1910 (1997)
31. Owen, A.B.: Scrambled net variance for integrals of smooth functions. Ann. Stat. **25**(4), 1541–1562 (1997)
32. Owen, A.B.: Scrambling Sobol' and Niederreiter-Xing points. J. Complex. **14**(4), 466–489 (1998)
33. Owen, A.B.: Local antithetic sampling with scrambled nets. Ann. Stat. **36**(5), 2319–2343 (2008)
34. Owen, A.B.: A constraint on extensible quadrature rules. Numerische Mathematik 1–8 (2015)
35. Owen, A.B.: Statistically efficient thinning of a Markov chain sampler. J. Comput. Graph. Stat. **26**(3), 738–744 (2017)
36. Owen, A.B.: Monte Carlo Book: The Quasi-Monte Carlo Parts (2019). https://statweb.stanford.edu/~owen/mc/
37. Owen, A.B., Rudolf, D.: A strong law of large numbers for scrambled net integration. SIAM Rev. (2020). To appear
38. Pan, Z., Owen, A.B.: The nonzero gain coefficients of Sobol's sequences are always powers of two. Technical report. Stanford University (2021). arXiv:2106.10534
39. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. **32**, 8026–8037 (2019)
40. Schürer, R., Schmid, W.C.: MinT-new features and new results. In: L'Ecuyer, P., Owen, A.B. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2008, pp. 501–512. Springer, Berlin (2009)
41. Sloan, I.H., Joe, S.: Lattice Methods for Multiple Integration. Oxford Science Publications, Oxford (1994)
42. Sobol', I.M.: The distribution of points in a cube and the accurate evaluation of integrals. USSR Comput. Math. Math. Phys. **7**(4), 86–112 (1967)
43. Sobol', I.M.: Multidimensional Quadrature Formulas and Haar Functions. Nauka, Moscow (1969). (In Russian)

44. Sobol', I.M.: Asymmetric convergence of approximations of the Monte Carlo method. Comput. Math. Math. Phys. **33**(10), 1391–1396 (1993)
45. Sobol', I.M., Asotsky, D., Kreinin, A., Kucherenko, S.: Construction and comparison of high-dimensional Sobol' generators. Wilmott Mag. **2011**(56), 64–79 (2011)
46. Surjanovic, S., Bingham, D.: Virtual library of simulation experiments: test functions and datasets (2013). https://www.sfu.ca/~ssurjano/
47. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Vijaykumar, A., Bardelli, A.P., Rothberg, A., Hilboll, A., Kloeckner, A., Scopatz, A., Lee, A., Rokem, A., Woods, C.N., Fulton, C., Masson, C., Häggström, C., Fitzgerald, C., Nicholson, D.A., Hagen, D.R., Pasechnik, D.V., Olivetti, E., Martin, E., Wieser, E., Silva, F., Lenders, F., Wilhelm, F., Young, G., Price, G.A., Ingold, G.L., Allen, G.E., Lee, G.R., Audren, H., Probst, I., Dietrich, J.P., Silterra, J., Webber, J.T., Slavič, J., Nothman, J., Buchner, J., Kulick, J., Schönberger, J.L., de Miranda Cardoso, J., Reimer, J., Harrington, J., Rodríguez, J.L.C., Nunez-Iglesias, J., Kuczynski, J., Tritz, K., Thoma, M., Newville, M., Kümmerer, M., Bolingbroke, M., Tartre, M., Pak, M., Smith, N.J., Nowaczyk, N., Shebanov, N., Pavlyk, O., Brodtkorb, P.A., Lee, P., McGibbon, R.T., Feldbauer, R., Lewis, S., Tygier, S., Sievert, S., Vigna, S., Peterson, S., More, S., Pudlik, T., Oshima, T., Pingel, T.J., Robitaille, T.P., Spura, T., Jones, T.R., Cera, T., Leslie, T., Zito, T., Krauss, T., Upadhyay, U., Halchenko, Y.O., Vázquez-Baeza, Y.: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods **17**(3), 261–272 (2020)
48. Yue, R.X., Mao, S.S.: On the variance of quadrature over scrambled nets and sequences. Stat. Probab. Lett. **44**(3), 267–280 (1999)

On the Distribution of Scrambled $(0, m, s)-$Nets Over Unanchored Boxes

Christiane Lemieux and Jaspar Wiart

Abstract We introduce a new quality measure to assess randomized low-discrepancy point sets of finite size n. This new quality measure, which we call "pairwise sampling dependence index", is based on the concept of negative dependence. A negative value for this index implies that the corresponding point set integrates the indicator function of any unanchored box with smaller variance than the Monte Carlo method. We show that scrambled $(0, m, s)-$nets have a negative pairwise sampling dependence index. We also illustrate through an example that randomizing via a digital shift instead of scrambling may yield a positive pairwise sampling dependence index.

Keywords Negative dependence · Scrambled nets · Discrepancy

1 Introduction

The quality of point sets used within quasi-Monte Carlo (QMC) methods is often assessed using the notion of discrepancy. For a point set $P_n = \{\mathbf{u}_i : i = 1, \ldots, n\}$, its star-discrepancy is given by $D_n^*(P_n) = \sup_{A \in \mathcal{A}_0} |J_n(A) - \mathrm{Vol(A)}|$ where $\mathcal{A}_0$ is the set of all boxes $A \subseteq [0, 1)^s$ anchored at the origin, and $J_n(A) = \sum_{i=1}^n \mathbf{1}_{\mathbf{u}_i \in A}/n$. The extreme discrepancy is instead given by $D_n(P_n) = \sup_{A \in \mathcal{A}} |J_n(A) - \mathrm{Vol(A)}|$ where $\mathcal{A}$ is the set of all boxes in $[0, 1)^s$. Both quantities are typically interpreted as comparing the empirical distribution induced by P_n with the uniform distribution over $[0, 1)^s$ in terms of the probability they assign to a given set $\mathcal{A}$ of boxes. Using inclusion-exclusion arguments, one can derive the bound $D_n(P_n) \leq 2^s D_n^*(P_n)$.

Many asymptotic results for $D_n^*(P_n)$ and $D_n(P_n)$ have been derived for various low-discrepancy sequences [1, 6]. These sequences are understood to be such that

C. Lemieux (✉)
University of Waterloo, 200 University Ave. West ON, Waterloo, ON N2L 3G1, Canada
e-mail: clemieux@uwaterloo.ca

J. Wiart
Johannes Kepler University, Altenbergerstr. 69, Linz, Upper Austria 4040, Austria
e-mail: jaspar.wiart@jku.at

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_5

$D_n^*(P_n) \in O((\log n)^s/n)$, and the above mentioned results often focus on studying the constant terms in the big-Oh notation and how it behaves as a function of s.

In practice, when using QMC methods, one is often working in settings where n is not too large, and a primary goal is to make sure that the QMC approximation will result in a better approximation than the one that would be obtained by using plain Monte Carlo sampling. One is also typically interested in assessing the approximation error, something naturally embedded in Monte Carlo methods via variance estimates and the central limit theorem.

In this setting, the use of randomized QMC methods is very appealing, as it preserves the advantage of QMC induced by the use of low-discrepancy sequences, while at the same time allowing for error estimates through independent and identically distributed (iid) replications.

In this paper, we focus on the above settings, i.e., where one (1) works with n not too large; (2) uses randomized QMC, and (3) hopes to do better than Monte Carlo.

To this end, we propose to reinterpret the measures $D_n^*(P_n)$ and $D_n(P_n)$ and propose a new, related measure that is designed for our chosen setting, which we refer to as "pairwise sampling dependence index". While this measure is meant to assess the uniformity of point sets much like the star and extreme discrepancies do, it also has another interpretation, which is that a point set with negative pairwise sampling dependence index estimates the expected value of the indicator function $\mathbf{1}_A$ for any $A \in \mathcal{A}$ with variance smaller than the Monte Carlo method. This new measure is defined in Sect. 2, Eq. (1). In Sect. 3 we revisit the result from [10], which shows that scrambled $(0, m, s)-$nets have a negative pairwise sampling dependence index over all anchored boxes. In Sect. 5 we show that this result extends to unanchored boxes in Theorem 4, which is the main result of this paper. Hence the extension to unanchored boxes does not cause the same deterioration of the bound for this uniformity measure as is the case when applying an inclusion-exclusion argument to go from the star to the extreme discrepancy.

The proof of this result is essentially a very difficult problem in linear programming, (something that is rather obscured by the fact that, since the number of variables depends on m, we work in $\ell^1(\mathbb{N})$ and its dual $\ell^\infty(\mathbb{N})$ rather than a finite dimensional space). Indeed, we must demonstrate that (1) is always negative for scrambled $(0, m, s)$-nets. We see in Theorem 2 that (1) is actually a linear equation whose variables are non-negative and are further constrained, in the one-dimensional case, according to Lemma 2. The constraints define a convex region whose extreme points, in the one-dimensional case, are found in Theorem 1 and given by (10) in the higher dimensional case. The remainder of the proof boils down to proving that (1) is non-negative at these extreme points. This requires the use of several technical lemmas (given in the appendix) proving sufficiently tight bounds on various combinatorial sums, which is precisely why we do not have to rely on an inclusion-exclusion argument to go from the anchored case to the unanchored one. We briefly discuss in Sect. 6 the advantage of scrambling over simpler randomization methods such as a digital shift. Ideas for future work are presented in Sect. 7.

2 Pairwise Sampling Dependence

We start by revisiting the definition of extreme discrepancy using a probabilistic approach, despite the fact that the point set P_n may be deterministic. We do so by introducing a quality measure we call *sampling discrepancy*, given by $\mathcal{D}_n(P_n) := \sup_{A\in\mathcal{A}} |\mathcal{P}_n(A) - \text{Vol}(A)|$, where $\mathcal{P}_n(A)$ is the probability that a randomly chosen point in P_n will fall in A. For a deterministic point set, this probability is given by $\mathcal{P}_n(A) = J_n(A)/n$, and so in this case $\mathcal{D}_n(P_n) = D_n(P_n)$. This definition captures how the discrepancy is often described as a distance measure between the empirical distribution induced by the point set P_n and the uniform distribution. Since the uniform distribution is viewed as a *target distribution* in this setting, we want this distance to be as small as possible.

In this paper we are interested in randomized QMC point sets $\tilde{P}_n$. We assume $\tilde{P}_n$ is a valid sampling scheme, meaning that $\mathbf{U}_i \sim U(0,1)^s$ for each $\mathbf{U}_i \in \tilde{P}_n$, with possibly some dependence among the $\mathbf{U}_i$'s. When we write $\tilde{P}_n$, we are thus not referring to a specific realization of the randomization process, which we instead denote by $\tilde{P}_n(\omega)$, where $\omega \in \Omega$, and Ω is the sampling space associated with our randomization process for P_n.

In that setting, we could compute $\mathcal{D}_n(\tilde{P}_n(\omega))$ and then perhaps compute the expected value of $\mathcal{D}_n(\tilde{P}_n(\omega))$ over all these realizations ω, or the probability that it will be larger than some value, as done in [3], for example. If we instead interpret $\mathcal{P}_n(A)$ as the probability that a randomly chosen point $\mathbf{U}_i$ from $\tilde{P}_n$ falls in A, then we would get $\mathcal{D}_n(\tilde{P}_n) = 0$, which is of little use.

To define an interesting alternative measure of uniformity for randomized QMC point sets, we introduce instead a "second-moment" version of the sampling discrepancy, in which we consider pairs of points rather than single points, and where the distribution against which we compare the point set is that induced by random sampling, where points are sampled independently from one another. When considering pairs of points, our goal is to examine the propensity for points to repel each other, which is a desirable feature if we want to achieve greater uniformity than random sampling. Note that this notion of "propensity to repel" is in line with the concept of negative dependence.

More precisely, we want to verify that the pairs of points from $\tilde{P}_n$ are less likely to fall within the same box A than they would if they were independent. Note that here, as was the case with $\mathcal{D}_n(P_n)$, we are comparing the distribution induced by $\tilde{P}_n$ with another distribution. But rather than comparing to a target distribution to which we want to be as close as possible, we are comparing to a distribution upon which we want to improve, and thus are not trying to be close to that distribution.

The measure we propose to assess the quality of a point set via the behavior of its pairs is called *pairwise sampling dependence index* and is given by

$$\mathcal{E}_n(\tilde{P}_n) := \sup_{A\in\mathcal{A}} H_n(A) - \text{Vol}^2(A), \tag{1}$$

$$\text{where the probability } H_n(A) := P((\mathbf{U}, \mathbf{V}) \in A \times A), \tag{2}$$

with $\mathbf{U}$ and $\mathbf{V}$ being distinct points randomly chosen from $\tilde{P}_n$. We say $\tilde{P}_n$ has a *negative pairwise sampling dependence index* when $\mathcal{E}_n(\tilde{P}_n) \le 0$. (This terminology is consistent with other measures of negative dependence: see, e.g., [10].)

Note that because we are not taking the supremum over all products of the form $A \times B$ with $A, B \in \mathcal{A}$ and instead only consider $A \times A$, it is possible, if $\tilde{P}_n$ is designed so that points tend to cluster away from each other, that the probability $H_n(A)$ will never be larger than what it is under random sampling, as given by $\mathrm{Vol}^2(A)$. This is not the case with the measure $\mathcal{D}_n(P_n)$, where having $\mathcal{P}_n(A) < \mathrm{Vol}(A)$ implies there will be some A' for which $\mathcal{P}_n(A') > \mathrm{Vol}(A')$. There the goal is to show there exist point sets P_n with $|\mathcal{P}_n(A) - \mathrm{Vol}(A)|$ very close to 0, and becoming closer to 0 as n goes to infinity. In our case, we instead want to show, for a given n, that there exist sampling schemes $\tilde{P}_n$ with $\mathcal{E}_n(\tilde{P}_n) \le 0$.

So far we mentioned the work done in [3, 10], but concepts of dependence based on measures different from (1) have recently been used in other works to analyze lattices [11], Latin hypercube sampling [4] and scrambled nets [2].

3 Revisiting Pairwise Sampling Dependence Over Anchored Boxes

In what follows, we assume $\tilde{P}_n$ is a scrambled $(0, m, s)-$net in base $b \ge s$, where $n = b^m$, and P_n represents the underlying $(0, m, s)-$net being scrambled. We assume the reader is familiar with the concept of digital nets and $(t, m, s)-$nets, as presented in e.g., [1, 6]. Also, when referring to scrambled nets, we refer to the scrambling method studied in [10], which originates from [8].

In [10], it was shown that if we restrict $\mathcal{E}_n(\tilde{P}_n)$ to anchored boxes—denote this version of $\mathcal{E}_n$ by $\mathcal{E}_{n,0}$—then $\mathcal{E}_{n,0}(\tilde{P}_n) \le 0$. In fact, a stronger result is shown in [10], which is that for $(\mathbf{U}, \mathbf{V})$ a pair of distinct points randomly chosen from $\tilde{P}_n$,

$$P((\mathbf{U}, \mathbf{V}) \in [\mathbf{0}, \mathbf{x}) \times [\mathbf{0}, \mathbf{y})) \le \mathrm{Vol}([\mathbf{0}, \mathbf{x}) \times [\mathbf{0}, \mathbf{y})), \qquad \text{for any } \mathbf{x}, \mathbf{y} \in [0, 1]^s.$$

For simplicity, in what follows we assume $\mathbf{x} = \mathbf{y}$ and let $A = [\mathbf{0}, \mathbf{x})$.

Next, to define a key quantity called the *volume vector* of a subset of $[0, 1]^{2s}$, we first define the regions $D_{\boldsymbol{i}}^s := \{(\mathbf{x}, \mathbf{y}) \in [0, 1)^{2s} : \boldsymbol{\gamma}_b^s(\mathbf{x}, \mathbf{y}) = \boldsymbol{i}\}$, where $\boldsymbol{\gamma}_b^s(\mathbf{x}, \mathbf{y}) := (\gamma_b(x_1, y_1), \ldots, \gamma_b(x_s, y_s))$ and $\gamma_b(x, y) \ge 0$ is the unique number $i \ge 0$ such that

$$\lfloor b^i x \rfloor = \lfloor b^i y \rfloor \quad \text{but} \quad \lfloor b^{i+1} x \rfloor \ne \lfloor b^{i+1} y \rfloor. \tag{3}$$

That is, $\gamma_b(x, y)$ is the exact number of initial digits shared by x and y in their base b expansion. If $x = y$ then we let $\gamma_b(x, y) = \infty$. Also, (3) implies $\gamma_b(x, y)$ is well defined for any $x, y \in [0, 1)$ even if x, y do not have a unique expansion in base b.

Let $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$. We can now define, for $A, B \subseteq [0, 1]^s$, the volume vector $V(A \times B) \in \ell^1(\mathbb{N}_0^s)$, whose component $V_{\boldsymbol{i}}(A \times B)$ associated to $\boldsymbol{i} \in \mathbb{N}_0^s$ is given by

$$V_{\boldsymbol{i}}(A \times B) := \int_{A\times B} \mathbf{1}_{D_{\boldsymbol{i}}^s} d\mathbf{u} d\mathbf{v} = \mathrm{Vol}((A \times B) \cap D_{\boldsymbol{i}}^s) \in [0, 1].$$

A key step used in [10] to prove that $H_n(A) \le \mathrm{Vol}(A \times A)$ is to find a conical combination of products of the form $\mathbf{1}_{\boldsymbol{k}} \times \mathbf{1}_{\boldsymbol{k}}$, where $\mathbf{1}_{\boldsymbol{k}} = \prod_{j=1}^s [0, b^{-k_j})$, $k_j \in \mathbb{N}_0$, $j = 1, \ldots, s$, whose volume vector is the same as that of $A \times A$. More precisely, one can find coefficients $t_{\boldsymbol{k}} \ge 0$ with $\sum_{\boldsymbol{k}\ge\mathbf{0}} t_{\boldsymbol{k}} = \mathrm{Vol}(A \times A)$, such that

$$V_{\boldsymbol{i}}(A \times A) = \sum_{\boldsymbol{k}\ge\mathbf{0}} t_{\boldsymbol{k}} b^{2|\boldsymbol{k}|} V_{\boldsymbol{i}}(\mathbf{1}_{\boldsymbol{k}} \times \mathbf{1}_{\boldsymbol{k}}) \qquad \text{for all } \boldsymbol{i} \in \mathbb{N}_0^s. \tag{4}$$

The coefficients $t_{\boldsymbol{k}}$ are shown in [10] to be given by $t_{\boldsymbol{k}} = \prod_{j=1}^s t_{k_j}$, where

$$t_k = \begin{cases} \frac{bV_k(A\times A) - V_{k-1}(A\times A)}{b-1} & \text{if } k > 0 \\ \frac{bV_0(A\times A)}{b-1} & \text{if } k = 0. \end{cases} \tag{5}$$

From here, rather than following the proof in [10], we exploit the fact that the joint pdf of points $(\mathbf{U}, \mathbf{V})$ from a scrambled $(0, m, s)-$net is a simple function that is constant on the $D_{\boldsymbol{i}}$ regions. The volume vector simply keeps track of how much the $D_{\boldsymbol{i}}$ region is covered by $A \times A$, allowing $H_n(A)$ to be written as a linear sum. Since the joint pdf is a simple function, these sums always have finitely many non-zero terms. See [10, Sect. 2.3] for more details.

Lemma 1 *Let $A = [\mathbf{0}, \mathbf{x})$ with $\mathbf{x} \in [0, 1]^s$, and let $t_{\boldsymbol{k}}$ be the coefficients for which (4) holds. Then for a scrambled $(0, m, s)-$net $\tilde{P}_n$*

$$H_n(A) = \sum_{\boldsymbol{k}\ge\mathbf{0}} t_{\boldsymbol{k}} b^{2|\boldsymbol{k}|} H_n(\mathbf{1}_{\boldsymbol{k}}). \tag{6}$$

Proof As shown in [10], we use the (constant) value $\psi_{\boldsymbol{i}}$ of the joint pdf of $\mathbf{U}, \mathbf{V}$ from $\tilde{P}_n$ over $D_{\boldsymbol{i}}^s$ to compute $H_n(A)$ as

$$H_n(A) = \sum_{\boldsymbol{i}\ge\mathbf{0}} \psi_{\boldsymbol{i}} \times V_{\boldsymbol{i}}(A \times A) \tag{7}$$

and then use (4) to get $H_n(A) = \sum_{\boldsymbol{i}\ge\mathbf{0}} \psi_{\boldsymbol{i}} \sum_{\boldsymbol{k}\ge\mathbf{0}} t_{\boldsymbol{k}} b^{2|\boldsymbol{k}|} V_{\boldsymbol{i}}(\mathbf{1}_{\boldsymbol{k}} \times \mathbf{1}_{\boldsymbol{k}}) = \sum_{\boldsymbol{k}\ge\mathbf{0}} t_{\boldsymbol{k}} b^{2|\boldsymbol{k}|} H_n(\mathbf{1}_{\boldsymbol{k}})$ where the order of summation can be changed thanks to Tonelli's theorem. □

Next, rather than computing $H_n(\mathbf{1}_{\boldsymbol{k}})$ by writing it as an integral involving the joint pdf associated with $(\mathbf{U}, \mathbf{V})$ (as we just did in the proof of Lemma 1), we instead use a conditional probability argument that allows us to directly connect this probability to the counting numbers $m_b(\boldsymbol{k}; P_n)$ used in [10], which for P_n a digital net, represents the number of points $\mathbf{u}_j \in P_n$ satisfying $\boldsymbol{\gamma}_b^s(\mathbf{u}_l, \mathbf{u}_j) \ge \boldsymbol{k}$ for a given $l \ne j$. (For an

arbitrary P_n, this number depends on ℓ but for a $(0, m, s)-$net, it is invariant with ℓ, hence we drop the dependence on ℓ in our notation. Also, since $m_b(\boldsymbol{k}; P_n) = m_b(\boldsymbol{k}; \tilde{P}_n)$, we work with the deterministic point sets when using these counting numbers.) This is a key step, as it allows us to write $H_n(A)$ as a linear equation instead of an integral, thereby yielding a linear programming formulation for our main result, which is to show $H_n(A) \leq \text{Vol}(A \times A)$. Specifically, we write

$$H_n(\mathbf{1}_{\boldsymbol{k}}) = P(\mathbf{V} \in \mathbf{1}_{\boldsymbol{k}} | \mathbf{U} \in \mathbf{1}_{\boldsymbol{k}}) P(\mathbf{U} \in \mathbf{1}_{\boldsymbol{k}}) = \frac{m_b(\boldsymbol{k}; P_n)}{n-1} b^{-|\boldsymbol{k}|}. \tag{8}$$

If P_n is a $(0, m, s)-$net in base b, then $m_b(\boldsymbol{k}; P_n) = \max(b^{m-|\boldsymbol{k}|} - 1, 0)$ [10]. These counting numbers are also closely connected to the key quantities [10]

$$C_b(\boldsymbol{k}; P_n) = \frac{b^{|\boldsymbol{k}|} m_b(\boldsymbol{k}; P_n)}{n-1}.$$

Combining (6) and (8), we get that for $\tilde{P}_n$ a scrambled $(0, m, s)-$net,

$$H_n(A) = \sum_{\boldsymbol{k} \geq \mathbf{0}} t_{\boldsymbol{k}} b^{|\boldsymbol{k}|} \frac{m_b(\boldsymbol{k}; P_n)}{n-1} = \sum_{\boldsymbol{k} \geq \mathbf{0}} t_{\boldsymbol{k}} C_b(\boldsymbol{k}; P_n) \leq \sum_{\boldsymbol{k} \geq \mathbf{0}} t_{\boldsymbol{k}} = \text{Vol}(A \times A), \tag{9}$$

since $C_b(\boldsymbol{k}; P_n) \leq 1$ when P_n is a $(0, m, s)-$net [10].

4 Decomposing Unanchored Intervals

We now consider the case where A is an unanchored box of the form $A = \prod_{j=1}^{s} [a_j, A_j)$, with $0 \leq a_j < A_j \leq 1$, $j = 1, \ldots, s$. In Sect. 5, we will prove in Theorem 3 that for a scrambled $(0, m, s)-$net, we still have $H_n(A) \leq \text{Vol}(A \times A)$ in this case, which is the main result of this paper. The proof of this result is much more difficult than in the anchored case because when A is not anchored at the origin, we cannot always find a conical decomposition of products of elementary intervals as in (4) that has the same volume vector as $A \times A$.

Before going further, we note that it is sufficient to focus on the decomposition of one-dimensional intervals A since a box is just a product of intervals. Hence for the rest of this section, we assume $s = 1$.

The reason why the decomposition (4) cannot be used for unanchored intervals is that it may produce coefficients $t_{\boldsymbol{k}}$ that are negative, which makes the inequality in (9) not necessarily true. In turn, this happens because the key property $bV_i(A \times A) \geq V_{i-1}(A \times A)$ that holds for an anchored interval A and that is used to show that $t_{\boldsymbol{k}} \geq 0$ in [10] is not always satisfied when A is an unanchored interval. In this case, the volume vector corresponding to $A \times A$ may be such that $V_0(A \times A) > 0$,

$V_1(A \times A) = \cdots = V_{r-1}(A \times A) = 0$, $V_i(A \times A) > 0$ for $i \geq r$. Because of this, we can see from (5) that some t_k may be negative.

To get a decomposition with non-negative coefficients, we introduce a family of regions of the form $Y \times Y$ where Y is not an elementary interval anchored at the origin. More precisely, for d, k non-negative integers, we define what we call an *elementary unanchored* $(d, k)-$*interval*

$$Y_k^{(d)} := \left[\frac{1}{b^{d+1}} - \frac{1}{b^{2+k+d}}, \frac{1}{b^{d+1}} + \frac{1}{b^{2+k+d}} \right).$$

As a first step, in the following lemma we establish some key properties for the volume vector corresponding to an unanchored interval A. It is the counterpart to the property that $bV_i(A \times A) \geq V_{i-1}(A \times A)$ for anchored boxes, and shows that the $V_i(A \times A)$'s do not decrease too quickly with i in the unanchored case, which is essential to prove the decomposition given in Theorem 1. The proof of this lemma is in the appendix. Note that this lemma applies to half-open intervals strictly contained in $[0, 1)$; the interval $[0, 1)$ can be handled using the decomposition from [10], which was described in the previous section.

Lemma 2 *Let $A \subset [0, 1)$ be a half-open interval and let $r \geq 1$ be the smallest integer such that we can write $A = [hb^{-r+1} + gb^{-r} - z, hb^{-r+1} + Gb^{-r} + Z)$ with $0 \leq h < b$, $1 \leq g \leq G \leq b - 1$ and $z, Z \in [0, b^{-r})$. Then $V(A \times A)$ is such that:*

1. *$V_i(A \times A) = 0$ for $i = 0, \ldots, r - 2$;*
2. *$bV_{i+1}(A \times A) \geq V_i(A \times A)$ for all $i \geq r$;*
3. *$V_{r-1}(A \times A) - \frac{b(b-2)}{b-1} V_r(A \times A) \leq \tilde{V}_r(A \times A)$, where $\tilde{V}_r(A \times A) = \sum_{i=r}^{\infty} V_i(A \times A)$.*

The next result establishes that any unanchored interval A in $[0, 1)$ has a volume vector $V(A \times A)$ that can be decomposed into a conical combination of volume vectors of elementary unanchored (d, k)-intervals $Y_k^{(d)}$ and elementary (anchored) $k-$intervals 1_k. Its proof is in the appendix.

Theorem 1 *Let $A \subseteq [0, 1)$ be a half-open interval. For $A \neq [0, 1)$, let $r \geq 1$ be the smallest positive integer such that we can write $A = [hb^{-r+1} + gb^{-r} - z, hb^{-r+1} + Gb^{-r} + Z)$ with $0 \leq h < b$, $1 \leq g \leq G \leq b - 1$, and $z, Z \in [0, b^{-r})$. For $A = [0, 1)$, let $r = 1$. Then there exists non-negative coefficients $(\alpha_k)_{k \geq 0}$ and $(\tau_k)_{k \geq 0}$ such that* $\mathrm{Vol}(A \times A) = \sum_{k \geq 0} (\alpha_k + \tau_k)$ *and*

$$V(A \times A) = \sum_{k=0}^{\infty} \alpha_k \frac{b^{2(k+r+1)}}{4} V(Y_k^{(r-1)} \times Y_k^{(r-1)}) + \sum_{k=0}^{\infty} \tau_k b^{2k} V(1_k \times 1_k).$$

5 Pairwise Sampling Dependence of Scrambled $(0, m, s)-$Nets on Unanchored Boxes

This section contains our main result, which is that scrambled $(0, m, s)-$nets have a negative pairwise sampling dependence index. That is, for this construction, $H_n(A) \leq \mathrm{Vol}(A \times A)$ for any unanchored box A. To prove this result, we must first provide a decomposition for $H_n(A)$ that makes use of elementary intervals and elementary unanchored $(d, k)-$intervals. To do so, we use the decomposition of an unanchored interval given in Theorem 1. First, we introduce some notation to denote regions in $[0, 1)^{2s}$ that will be used repeatedly in this section, starting with those we get from the decomposition proved in Theorem 1:

$$D(\boldsymbol{k}, \boldsymbol{d}, J) := \prod_{j \in J} Y_{k_j}^{(d_j)} \times Y_{k_j}^{(d_j)} \prod_{j \in J^c} 1_{k_j} \times 1_{k_j}, \tag{10}$$

where $J \subseteq \{1, \ldots, s\}$. The interval $Y_{k_j}^{(d_j)}$ is decomposed further using

$$Y_{k_j,1}^{(d_j)} := \left[\frac{1}{b^{d_j+1}} - \frac{1}{b^{2+k_j+2_j}}, \frac{1}{b^{d_j+1}} \right), \text{ and } Y_{k_j,2}^{(d_j)} := \left[\frac{1}{b^{d_j+1}}, \frac{1}{b^{d_j+1}} + \frac{1}{b^{2+k_j+2_j}} \right).$$

We also make use of the following sub-regions, where $I, K \subseteq J$:

$$\begin{aligned}
E(\boldsymbol{k}, \boldsymbol{d}, J, I) &:= \prod_{j \in I} 1_{k_j+d_j+2} \times 1_{k_j+d_j+2} \prod_{j \in J^c} 1_{k_j} \times 1_{k_j} \\
\tilde{E}(\boldsymbol{k}, \boldsymbol{d}, J, I, K) &:= \prod_{j \in I \cap K} Y_{k_j,2}^{(d_j)} \times Y_{k_j,2}^{(d_j)} \prod_{j \in I \cap K^c} Y_{k_j,1}^{(d_j)} \times Y_{k_j,1}^{(d_j)} \prod_{j \in J^c} 1_{k_j} \times 1_{k_j} \\
F(\boldsymbol{k}, \boldsymbol{d}, J, I) &:= E(\boldsymbol{k}, \boldsymbol{d}, J, I) \times \prod_{j \in J \cap I^c} Y_{k_j,1}^{(d_j)} \times Y_{k_j,2}^{(d_j)} \\
F(\boldsymbol{k}, \boldsymbol{d}, J, I, K) &:= \tilde{E}(\boldsymbol{k}, \boldsymbol{d}, J, I, K) \prod_{j \in J \cap I^c \cap K} Y_{k_j,1}^{(d_j)} \times Y_{k_j,2}^{(d_j)} \prod_{j \in J \cap I^c \cap K^c} Y_{k_j,2}^{(d_j)} \times Y_{k_j,1}^{(d_j)}.
\end{aligned}$$

The region $F(\boldsymbol{k}, \boldsymbol{d}, J, I)$ in which a pair of points $(\mathbf{U}, \mathbf{V})$ lies will sometimes be written as the product of the two regions obtained by projecting it over the coordinates of $\mathbf{U}$ and then $\mathbf{V}$, using the notation $F(\boldsymbol{k}, \boldsymbol{d}, J, I) = F_1(\boldsymbol{k}, \boldsymbol{d}, J, I) \times F_2(\boldsymbol{k}, \boldsymbol{d}, J, I)$. That is, $F_i(\boldsymbol{k}, \boldsymbol{d}, J, I) := \prod_{j \in I} 1_{k_j+d_j+2} \prod_{j \in J^c} 1_{k_j} \prod_{j \in J \cap I^c} Y_{k_j,i}^{(d_j)}$ for $i = 1, 2$.

Next, we define the counting numbers $m_b(\boldsymbol{k}, \boldsymbol{d}, c, J, I; P_n)$. The parameter $c \geq 0$ is used to specify the number of initial common digits over the subset I.

Definition 1 For P_n a digital net, let $m_b(\boldsymbol{k}, \boldsymbol{d}, c, J, I; P_n)$ be the number of points $\mathbf{u}_\ell$, for a given point $\mathbf{u}_i \in P_n$, which are different from $\mathbf{u}_i$ and satisfy:

$$\gamma_b(u_{i,j}, u_{\ell,j}) \geq k_j + d_j + c \text{ if } j \in I;$$
$$\gamma_b(u_{i,j}, u_{\ell,j}) \geq k_j \text{ if } j \in J^c;$$
$$\gamma_b(u_{i,j}, u_{\ell,j}) = d_j \text{ if } j \in J \cap I^c.$$

The properties stated in the next lemma involve the above regions and will be useful to prove Theorems 2 and 3. Its proof is in the appendix.

Lemma 3 *Let $|\boldsymbol{k}|_J$ denote the sum $\sum_{j\in J} k_j$ with also $|\boldsymbol{k}+2|_J = \sum_{j\in J}(k_j+2) = |\boldsymbol{k}|_J + 2|J|$. Then:*

1. $\text{Vol}(D(\boldsymbol{k}, \boldsymbol{d}, J)) = 2^{2|J|} b^{-2(|\boldsymbol{k}|+|\boldsymbol{d}+2|_J)}$.
2. $\text{Vol}(F_1(\boldsymbol{k}, \boldsymbol{d}, J, I)) = b^{-(|\boldsymbol{k}|+|\boldsymbol{d}+2|_J)}$.
3. $P(\mathbf{V} \in F_2(\boldsymbol{k}, \boldsymbol{d}, J, I) | \mathbf{U} \in F_1(\boldsymbol{k}, \boldsymbol{d}, J, I)) = \frac{m_b(\boldsymbol{k},\boldsymbol{d},2,J,I;P_n)}{n-1} \frac{(b-1)^{|I|-|J|}}{b^{|\boldsymbol{k}|_{J\cap J^c}+|J|-|I|}}$.
4. *We have* $D(\boldsymbol{k}, \boldsymbol{d}, J) = \cup_{K,I\subseteq J} F(\boldsymbol{k}, \boldsymbol{d}, J, I, K)$ *and* $P((\mathbf{U}, \mathbf{V}) \in F(\boldsymbol{k}, \boldsymbol{d}, J, I)) = P((\mathbf{U}, \mathbf{V}) \in F(\boldsymbol{k}, \boldsymbol{d}, J, I, K))$ *for all* $K, I \subseteq J$.

The next result provides us with a key decomposition for $H_n(A)$.

Theorem 2 *Let $A = \prod_{j=1}^s [a_j, A_j)$ be an unanchored box, where $0 \leq a_j < A_j \leq 1$, $j = 1, \ldots, s$. For $J \subseteq \{1, \ldots, s\}$ and $\boldsymbol{k} \in \mathbb{N}_0^s$, let $\tilde{\alpha}_{\boldsymbol{k},J} = \prod_{j\in J} \alpha_{k_j}^{(j)} \prod_{j\in J^c} \tau_{k_j}^{(j)}$, where the $\alpha_{k_j}^{(j)}$ and $\tau_{k_j}^{(j)}$ come from the decomposition given in Theorem 1 applied to the interval $[a_j, A_j)$, $j = 1, \ldots, s$. In particular, this means $\tilde{\alpha}_{\boldsymbol{k},J} \geq 0$ and*

$$\sum_{\boldsymbol{k}\geq \mathbf{0}} \sum_{J\subseteq\{1,\ldots,s\}} \tilde{\alpha}_{\boldsymbol{k},J} = \text{Vol}(A \times A). \tag{11}$$

Let $(\mathbf{U}, \mathbf{V})$ be a randomly chosen pair of points from a point set $\tilde{P}_n$. Then

$$H_n(A) = \sum_{\boldsymbol{k}\geq \mathbf{0}} \sum_{J\subseteq\{1,\ldots,s\}} \frac{\tilde{\alpha}_{\boldsymbol{k},J}}{\text{Vol}(D(\boldsymbol{k}, \boldsymbol{d}, J))} P((\mathbf{U}, \mathbf{V}) \in D(\boldsymbol{k}, \boldsymbol{d}, J)).$$

Proof Using Theorem 1 we can write

$$\begin{aligned}
V(A \times A) &= \prod_{j=1}^s \left(\sum_{k_j\geq 0} \alpha_{k_j}^{(j)} \frac{b^{2(k_j+d_j+2)}}{4} V(Y_{k_j}^{(d_j)} \times Y_{k_j}^{(d_j)}) + \sum_{k_j\geq 0} \tau_{k_j}^{(j)} b^{2k_j} V(1_{k_j} \times 1_{k_j}) \right) \\
&= \sum_{\boldsymbol{k}\geq \mathbf{0}} \sum_{J\subseteq\{1,\ldots,s\}} \left(\prod_{j\in J} \alpha_{k_j}^{(j)} \frac{b^{2(k_j+d_j+2)}}{4} \prod_{j\in J^c} b^{2k_j} \tau_{k_j}^{(j)} \right) V(D(\boldsymbol{k}, \boldsymbol{d}, J)) \\
&= \sum_{\boldsymbol{k}\geq \mathbf{0}} \sum_{J\subseteq\{1,\ldots,s\}} \tilde{\alpha}_{\boldsymbol{k},J} 2^{-2|J|} b^{2(|\boldsymbol{k}|+|\boldsymbol{d}+2|_J)} V(D(\boldsymbol{k}, \boldsymbol{d}, J)) \\
&= \sum_{\boldsymbol{k}\geq \mathbf{0}} \sum_{J\subseteq\{1,\ldots,s\}} \frac{\tilde{\alpha}_{\boldsymbol{k},J}}{\text{Vol}(D(\boldsymbol{k}, \boldsymbol{d}, J))} V(D(\boldsymbol{k}, \boldsymbol{d}, J)),
\end{aligned}$$

where the last equality follows from Part 1 of Lemma 3. Then, using the same kind of reasoning as in Lemma 1, we get

$$H_n(A) = \sum_{\boldsymbol{k}\geq\boldsymbol{0}} \sum_{J\subseteq\{1,\ldots,s\}} \frac{\tilde{\alpha}_{\boldsymbol{k},J}}{\mathrm{Vol}(D(\boldsymbol{k},\boldsymbol{d},J))} P((\mathbf{U},\mathbf{V}) \in D(\boldsymbol{k},\boldsymbol{d},J)).$$

□

It is clear from Theorem 2 that in order to prove that $H_n(A) \leq \mathrm{Vol}(A \times A)$, it is sufficient to prove that $P((\mathbf{U},\mathbf{V}) \in D(\boldsymbol{k},\boldsymbol{d},J)) \leq \mathrm{Vol}(D(\boldsymbol{k},\boldsymbol{d},J))$ for all $\boldsymbol{k},\boldsymbol{d},J$. That is, the regions $D(\boldsymbol{k},\boldsymbol{d},J)$ correspond to the extreme points in the linear programming formulation of our problem.

A key quantity to analyze this probability is the following weighted sum of counting numbers for P_n, where $J \subseteq \{1,\ldots,s\}$ and $I^* := I \cup J^c$:

$$\tilde{m}_b(\boldsymbol{k},\boldsymbol{d},J;P_n) := \frac{1}{2^{|J|}} \sum_{I\subseteq J} b^{|\boldsymbol{k}|_{I^*}+|\boldsymbol{d}|_J+|J|+|I|}(b-1)^{|I|-|J|} \frac{m_b(\boldsymbol{k},\boldsymbol{d},2,J,I;P_n)}{n-1}. \tag{12}$$

Theorem 3 *Let A be an unanchored box in $[0,1)^s$. Let $H_n(A)$ and $\tilde{\alpha}_{\boldsymbol{k},J}$ be defined as in Theorem 2. Let P_n have counting numbers $m_b(\boldsymbol{k},\boldsymbol{d},2,J,I;P_n)$ such that*

$$\tilde{m}_b(\boldsymbol{k},\boldsymbol{d},J;P_n) \leq 1, \tag{13}$$

where $\tilde{m}_b(\boldsymbol{k},\boldsymbol{d},J;P_n)$ is defined in (12). Then the scrambled point set $\tilde{P}_n$ is such that

$$H_n(A) \leq \sum_{\boldsymbol{k}\geq\boldsymbol{0}} \sum_{J\subseteq\{1,\ldots,s\}} \tilde{\alpha}_{\boldsymbol{k},J} = \mathrm{Vol}(A \times A).$$

Proof As mentioned earlier, based on Theorem 2, it suffices to show that $P((\mathbf{U},\mathbf{V}) \in D(\boldsymbol{k},\boldsymbol{d},J)) \leq \mathrm{Vol}(D(\boldsymbol{k},\boldsymbol{d},J))$ for all 5-tuples $(m,s,\boldsymbol{k},\boldsymbol{d},J)$, where $m \geq 1, s \geq 1, \boldsymbol{k} \geq \boldsymbol{0}, \boldsymbol{d} \geq \boldsymbol{0}, J \subseteq \{1,\ldots,s\}$. Indeed, if this holds, then from Theorem 2 and using the fact that $\tilde{\alpha}_{\boldsymbol{k},J} \geq 0$, we can derive the inequality

$$H_n(A) = \sum_{\boldsymbol{k}} \sum_{J} \tilde{\alpha}_{\boldsymbol{k},J} \frac{P((\mathbf{U},\mathbf{V}) \in D(\boldsymbol{k},\boldsymbol{d},J))}{\mathrm{Vol}(D(\boldsymbol{k},\boldsymbol{d},J))} \leq \sum_{\boldsymbol{k}} \sum_{J} \tilde{\alpha}_{\boldsymbol{k},J} = \mathrm{Vol}(A \times A),$$

where the last equality is obtained from (11), also proved in Theorem 2.

To analyze the probability $P((\mathbf{U},\mathbf{V}) \in D(\boldsymbol{k},\boldsymbol{d},J))$, we use the decomposition of $D(\boldsymbol{k},\boldsymbol{d},J)$ into the sub-regions $F(\boldsymbol{k},\boldsymbol{d},J,I)$ outlined in Part 4 of Lemma 3:

$$\frac{P((\mathbf{U},\mathbf{V}) \in D(\boldsymbol{k},\boldsymbol{d},J))}{\mathrm{Vol}(D(\boldsymbol{k},\boldsymbol{d},J))} = \sum_{I\subseteq J} 2^{|J|}\frac{P((\mathbf{U},\mathbf{V}) \in F(\boldsymbol{k},\boldsymbol{d},J,I))}{\mathrm{Vol}(D(\boldsymbol{k},\boldsymbol{d},J))}$$

$$= \frac{1}{\mathrm{Vol}(D(\boldsymbol{k},\boldsymbol{d},J))}\cdot$$

$$\cdot \sum_{I\subseteq J} 2^{|J|} P(\mathbf{U} \in F_1(\boldsymbol{k},\boldsymbol{d},J,I)) P(\mathbf{V} \in F_2(\boldsymbol{k},\boldsymbol{d},J,I)|\mathbf{U} \in F_1(\boldsymbol{k},\boldsymbol{d},J,I))$$

$$= \sum_{I\subseteq J} 2^{|J|}\frac{\mathrm{Vol}(F_1(\boldsymbol{k},\boldsymbol{d},J,I))}{\mathrm{Vol}(D(\boldsymbol{k},\boldsymbol{d},J))}\frac{m_b(\boldsymbol{k},\boldsymbol{d},2,J,I;P_n)}{n-1}\frac{(b-1)^{|I|-|J|}}{b^{|\boldsymbol{k}+1|_{J\cap I^c}}}$$

$$= \sum_{I\subseteq J} 2^{|J|}\frac{b^{-(|\boldsymbol{k}|+|\boldsymbol{d}+2|_J)}}{2^{2|J|}b^{-2(|\boldsymbol{k}|+|\boldsymbol{d}+2|_J)}}\frac{m_b(\boldsymbol{k},\boldsymbol{d},2,J,I;P_n)}{n-1}\frac{(b-1)^{|I|-|J|}}{b^{|\boldsymbol{k}+1|_{J\cap I^c}}}$$

$$= \sum_{I\subseteq J} \frac{1}{2^{|J|}} b^{|\boldsymbol{k}|_{I^*}+|\boldsymbol{d}|_J+|J|+|I|}(b-1)^{|I|-|J|}\frac{m_b(\boldsymbol{k},\boldsymbol{d},2,J,I;P_n)}{n-1}$$

$$= \tilde{m}_b(\boldsymbol{k},\boldsymbol{d},J;P_n) \le 1,$$

where the first equality comes from Lemma 3 (Part 4), the third from Lemma 3 (Part 3), the fourth from Lemma 3 (Parts 1, 2), and the last inequality follows from (13). □

To get to our ultimate goal—which is captured in Theorem 4 and is to prove that $H_n(A) \le \mathrm{Vol}(A \times A)$ for an unanchored box A for a scrambled $(0, m, s)-$net—thanks to Theorem 3 all we need to do is to show that the condition (13) indeed holds for a $(0, m, s)-$net. The rest of this section is devoted to this (cumbersome) task.

First we write $\tilde{m}_b(\boldsymbol{k},\boldsymbol{d},J;P_n) = \sum_I \psi_m(\boldsymbol{k},\boldsymbol{d},J,I)/2^{|I|}$, where

$$\psi_m(\boldsymbol{k},\boldsymbol{d},J,I) := b^{|\boldsymbol{k}|_{I^*}+|\boldsymbol{d}|_J+|J|+|I|}(b-1)^{|I|-|J|}\frac{m_b(\boldsymbol{k},\boldsymbol{d},2,J,I;P_n)}{n-1}. \tag{14}$$

The difficulty that arises when trying to bound the sum (12) by 1 is that some of the terms (14) can be larger than 1 for certain combinations of $m, \boldsymbol{k}, \boldsymbol{d}$, and J. Hence we need to show that the smaller terms compensate for those larger than 1 so that overall, the average of these terms is indeed bounded by 1.

Now, we will not work directly with the counting numbers $m_b(\boldsymbol{k},\boldsymbol{d},2,J,I;P_n)$ and will instead bound them, which in turn will yield a bound on $\psi_m(\boldsymbol{k},\boldsymbol{d},J,I)$ via (14) and thus a bound on $\tilde{m}_b(\boldsymbol{k},\boldsymbol{d},J;P_n)$. To show this bound is no larger than 1. we break the problem in different cases, depending on the relative magnitude of m vs. $|\boldsymbol{k}|$, $|\boldsymbol{d}|_J$ and $|J|$, with resulting bounds shown in Propositions 1, 2 and 3.

The bounds on the $\psi_m(\boldsymbol{k},\boldsymbol{d},I,J)$ terms will make use of the following functions.

Definition 2 Let ℓ, j, i be non-negative integers with $j > i$. We define

$$h_{j,i}(\ell) = \frac{b^{j+i-\ell}}{(b-1)^{j-i}}\binom{j-i-1}{\ell-2i}, \qquad 2i+1 < \ell < j+i, \qquad 0 \le i < j, \tag{15}$$

$$\text{and } g_{j,i}(\ell) = \begin{cases} 1 & \text{if } \ell \geq i+j \text{ or, if } \ell > 2i \text{ and } \ell \text{ is even} \\ 1 + h_{j,i}(\ell) & \text{if } 2i+1 < \ell < j+i \text{ and } \ell \text{ is odd} \\ \left(\frac{b}{b-1}\right)^{j-i-1} & \text{if } \ell = 2i+1 \\ 0 & \text{if } \ell \leq 2i. \end{cases} \tag{16}$$

In some cases, the following bound on $g_{i,j}(\ell)$ will be enough for our purpose. (Both Lemmas 4 and 5 are proved in the appendix.)

Lemma 4 *Let $j > i \geq 0$. Then*

$$g_{i,j}(\ell) \leq \left(\frac{b}{b-1}\right)^{i+j-\ell} \text{ when } 2i < \ell < i+j.$$

The next lemma gives a bound on $\psi_m(\boldsymbol{k}, \boldsymbol{d}, J, I)$ in the case of a $(0, m, s)-$net.

Lemma 5 *If P_n is a $(0, m, s)-$net, then for $I \subset J$, $\psi_m(\boldsymbol{k}, \boldsymbol{d}, J, I)$ satisfies*

$$\psi_m(\boldsymbol{k}, \boldsymbol{d}, J, I) \leq \frac{b^m}{b^m - 1} g_{|J|,|I|}(m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J). \tag{17}$$

Moreover, when $2|I| < m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J < |J| + |I|$, then

$$\psi_m(\boldsymbol{k}, \boldsymbol{d}, J, I) \leq \frac{b^m}{b^m - 1} \left(\frac{b-1}{b}\right)^{m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - |I| - |J|}. \tag{18}$$

Having found a bound for $\psi_m(\boldsymbol{k}, \boldsymbol{d}, J, I)$ for the possible ranges of values for m, we can now return to the task of bounding the weighted sum $\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n)$. We start with the easiest case.

Proposition 1 *Let P_n be a $(0, m, s)-$net. If $J \neq \emptyset$ and $m \geq |\boldsymbol{k}| + |\boldsymbol{d}|_J + 2|J|$ then $\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) \leq 1$.*

Proof In this case, $m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J \geq |I| + |J|$ for all I (for a given J) and therefore

$$\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) = \frac{1}{2^{|J|}} \sum_{I \subseteq J} \psi_m(\boldsymbol{k}, \boldsymbol{d}, J, I) \leq \frac{1}{2^{|J|}} \frac{b^m}{b^m - 1} \sum_{I \subset J} 1 + \frac{1}{2^{|J|}} \psi_m(\boldsymbol{k}, \boldsymbol{d}, J, J),$$

where the inequality is derived from Lemma 5. Observing that $m_b(\boldsymbol{k}, \boldsymbol{d}, 2, J, J; P_n) = m_b(\tilde{\boldsymbol{k}}; P_n)$, where $\tilde{k}_j := k_j$ if $j \in J^c$ and $\tilde{k}_j := k_j + d_j + 2$ if $j \in J$, we get

$$\psi_m(\boldsymbol{k}, \boldsymbol{d}, J, J) = b^{|\boldsymbol{k}| + |\boldsymbol{d}|_J + 2|J|} \frac{m_b(\tilde{\boldsymbol{k}}; P_n)}{n-1} = b^{|\boldsymbol{k}| + |\boldsymbol{d}|_J + 2|J|} \frac{b^{m - |\boldsymbol{k}| - |\boldsymbol{d}|_J - 2|J|} - 1}{n-1} \tag{19}$$

and therefore obtain

$$\begin{aligned}\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) &\le \frac{1}{2^{|J|}} \frac{b^m}{b^m - 1} \sum_{I \subset J} 1 + \frac{1}{2^{|J|}} b^{|\boldsymbol{k}|+|\boldsymbol{d}|_J+2|J|} \frac{b^{m-|\boldsymbol{k}|-|\boldsymbol{d}|_J-2|J|} - 1}{b^m - 1} \\ &= \frac{2^{|J|} - 1}{2^{|J|}} \frac{b^m}{b^m - 1} + \frac{1}{2^{|J|}} \frac{b^m - b^{|\boldsymbol{k}|+|\boldsymbol{d}|_J+2|J|}}{b^m - 1} \\ &= \frac{1}{2^{|J|}(b^m - 1)} \left(b^m 2^{|J|} - b^{|\boldsymbol{k}|+|\boldsymbol{d}|_J+2|J|}\right) \\ &= \frac{b^m}{b^m - 1} \left(\frac{2^{|J|} - b^{|\boldsymbol{k}|+|\boldsymbol{d}|_J+2|J|-m}}{2^{|J|}}\right).\end{aligned}$$

Therefore $\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) \le 1$ if $b^m 2^{|J|} - b^{|\boldsymbol{k}|+|\boldsymbol{d}|_J+2|J|} \le 2^{|J|}(b^m - 1)$, or equivalently, if $2^{|J|} \le b^{|\boldsymbol{k}|+|\boldsymbol{d}|_J+2|J|}$, which is true since $b \ge 2$, and $|\boldsymbol{k}| + |\boldsymbol{d}|_J + 2|J| > |J|$. □

Next, we deal with the more difficult case $m < |\boldsymbol{k}| + |\boldsymbol{d}|_J + 2|J|$, which implies that the bound given in Lemma 5 for $\psi_m(\boldsymbol{k}, \boldsymbol{d}, J, I)$ is sometimes larger than 1. Note that from (19), we see that $m_b(\tilde{\boldsymbol{k}}; P_n) = 0$ and thus $\psi_m(\boldsymbol{k}, \boldsymbol{d}, J, J) = 0$ in this case.

To handle this case, we need to analyze the function

$$G(m, s, J, \boldsymbol{k}, \boldsymbol{d}) = \sum_{I \subset J} g_{|J|,|I|}(m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J),$$

which we may at times write as

$$\begin{aligned}&G(m, s, J, \boldsymbol{k}, \boldsymbol{d}) \\ &= \sum_{I \subset J : m^* > 2|I|+1} 1 + \sum_{I \in \mathcal{M}(J)} h_{|J|,|I|}(m^*) + \sum_{I \subset J : m^* = 2|I|+1} \left(\frac{b}{b-1}\right)^{|J| - 0.5(m^*-1)-1}\end{aligned}$$

where $m^* := m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J$ and $\mathcal{M}(J) = \{I \subset J : 2|I| + 1 < m^* < |I| + |J|, m^* \text{odd}\}$.

To show $\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) = \sum_{I \subset J} \psi_m(\boldsymbol{k}, \boldsymbol{d}, J, I)/2^{|J|} \le 1$, from the bound (17) on $\psi_m(\boldsymbol{k}, \boldsymbol{d}, J, I)$ we see it is sufficient to show $G(m, s, J, \boldsymbol{k}, \boldsymbol{d}) \le 2^{|J|} \frac{b^m - 1}{b^m}$ since then

$$\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) \le \frac{1}{2^{|J|}} \frac{b^m}{b^m - 1} G(m, s, J, \boldsymbol{k}, \boldsymbol{d}) \le 1.$$

The following lemma will allow us to set $\boldsymbol{d} = \boldsymbol{0}$ when bounding $G(m, s, J, \boldsymbol{k}, \boldsymbol{d})$.

Lemma 6 *If $g_0 \ge 0$ is a constant such that $G(m, s, J, \boldsymbol{k}, \boldsymbol{0}) \le g_0$ for all $(m, s, J, \boldsymbol{k})$, then $G(m, s, J, \boldsymbol{k}, \boldsymbol{d}) \le g_0$ for all $(m, s, J, \boldsymbol{k}, \boldsymbol{d})$.*

Proof If $m < |\boldsymbol{d}|_J$ then $m^* < 2|I|$ for all $I \subset J$ and therefore $G(m, s, J, \boldsymbol{k}, \boldsymbol{d}) = 0$. If $m \ge |\boldsymbol{d}|_J$ then it is easy to see that $G(m, s, J, \boldsymbol{k}, \boldsymbol{d}) = G(m - |\boldsymbol{d}|_J, s, J, \boldsymbol{k}, \boldsymbol{0})$, because $(m, s, J, \boldsymbol{k}, \boldsymbol{d})$ and $(m - |\boldsymbol{d}|_J, s, J, \boldsymbol{k}, \boldsymbol{0})$ yield the same m^* for all $I \subset J$, and $G(m, s, J, \boldsymbol{k}, \boldsymbol{d})$ only depend on m, $\boldsymbol{k}$, and $\boldsymbol{d}$ through m^*. □

Based on this result, we set $\boldsymbol{d} = \mathbf{0}$ in what follows, and consider two different sub-cases. The respective bounds on $G(m, s, J, \boldsymbol{k}, \mathbf{0})$ are given in Propositions 2 and 3, which also establish that the condition (13)—stating that $\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) \leq 1$—holds for each sub-case. Before we state and prove these two propositions, we first state a technical lemma needed in the proof of Proposition 2, and proved in the appendix.

Lemma 7 *For $b \geq s \geq 2$ and $\tilde{s} = \lfloor \frac{s}{2} \rfloor - 1$. Then*

$$R(b, s) := \frac{1}{2^s} \sum_{j=0}^{\tilde{s}} \binom{s}{j} \left(\frac{b}{b-1} \right)^{s-j} \leq 1.$$

Proposition 2 *Let P_n be a $(0, m, s)$−net. If $J \neq \emptyset$ and $m < |J|$ then $G(m, s, J, \boldsymbol{k}, \mathbf{0}) \leq (b-1)/b$ for all $\boldsymbol{k}$ and therefore $\tilde{m}_b(\boldsymbol{k}, \mathbf{0}, J; P_n) \leq 1$.*

Proof The fact that $m < |J|$ implies $m - |\boldsymbol{k}|_{I^*} < |J| + |I|$ for all I. Also, if $|I| \geq 0.5(|J| - 1)$, then $m - |\boldsymbol{k}|_{I^*} \leq 2|I|$ for all $\boldsymbol{k}$ and then $g_{|J|,|I|}(m - |\boldsymbol{k}|_{I^*}) = 0$. Thus

$$G(m, s, J, \boldsymbol{k}, \mathbf{0}) \leq \sum_{I : |I| < 0.5(|J|-1)} g_{|J|,|I|}(m - |\boldsymbol{k}|_{I^*}).$$

It turns out that in this case, the simpler but larger bound (18) can be used (since the only non-zero $g_{|J|,|I|}(m - |\boldsymbol{k}|_{I^*})$ terms are those for which $2|I| < m - |\boldsymbol{k}|_{I^*} < |I| + |J|$, which means (18) can indeed be applied), so we have

$$\begin{aligned} & G(m, s, J, \boldsymbol{k}, \mathbf{0}) \\ & \leq \sum_{I : |I| < 0.5(|J|-1)} \left(\frac{b-1}{b} \right)^{m - |\boldsymbol{k}|_{I^*} - |J| - |I|} \leq \sum_{i=0}^{\lfloor 0.5|J| \rfloor - 1} \binom{s}{i} \left(\frac{b}{b-1} \right)^{|J| - i - 1} \\ & = \frac{b-1}{b} \sum_{i=0}^{\lfloor 0.5|J| \rfloor - 1} \binom{|J|}{i} \left(\frac{b}{b-1} \right)^{|J| - i}, \end{aligned} \tag{20}$$

where the second inequality comes from the fact that $m - |\boldsymbol{k}|_{I^*} > 2|I|$ implies $|J| + |I| + |\boldsymbol{k}|_{I^*} - m \leq |J| - |I| - 1$.

Using Lemma 7 with $s = |J|$, we get that the sum in (20) is bounded by 1. Hence

$$\tilde{m}_b(\boldsymbol{k}, \mathbf{0}, J; P_n) \leq \frac{1}{2^{|J|}} \frac{b^m}{b^m - 1} \frac{b-1}{b} < 1, \text{ for any } |J| \geq 2, m \geq 1.$$

□

The last case we need to deal with is when m is such that $|J| \leq m < |\boldsymbol{k}| + 2|J|$. Let $\mathcal{B}$ be the set of pairs $(m, \boldsymbol{k})$ satisfying this assumption.

To handle this case, we make use of the following two lemmas about $G(m, s, \boldsymbol{k}, J, \mathbf{0})$. The first one shows that when $\boldsymbol{k} = \boldsymbol{d} = \mathbf{0}$ the maximum is reached

when $m = 2|J| - 1$. The second one shows it is sufficient to bound $G(m, s, J, \boldsymbol{k}, \boldsymbol{0})$ at $\boldsymbol{k} = \boldsymbol{0}$.

Lemma 8 *If $\boldsymbol{k} = \boldsymbol{d} = \boldsymbol{0}$, then $G(m, s, J, \boldsymbol{0}, \boldsymbol{0}) \leq G(2|J| - 1, s, J, \boldsymbol{0}, \boldsymbol{0}) = 2^{|J|} - 1$ for all m such that $(m, \boldsymbol{0}) \in \mathcal{B}$.*

Lemma 9 *Consider a pair $(m, \boldsymbol{k})$ with possibly $\boldsymbol{k} \neq \boldsymbol{0}$. Then there exists an odd integer value $\tilde{m}$ such that $G(\tilde{m}, s, J, \boldsymbol{0}, \boldsymbol{0}) \geq G(m, s, J, \boldsymbol{k}, \boldsymbol{0})$.*

Using these two lemmas (proved in the appendix), we get a bound on $G(m, s, J, \boldsymbol{k}, \boldsymbol{0})$ for this last case, which in turn allows us to show that (13) also holds then.

Proposition 3 *Let P_n be a $(0, m, s)$−net. Assume m is such that $|J| \leq m < |\boldsymbol{k}| + 2|J|$. Then $G(m, s, J, \boldsymbol{k}, \boldsymbol{0}) \leq 2^{|J|} - 1$ and therefore $\tilde{m}_b(\boldsymbol{k}, \boldsymbol{0}, J; P_n) \leq 1$.*

Proof For a given s and J, we need to find a bound for $G(m, s, J, \boldsymbol{k}, \boldsymbol{0})$ over all pairs $(m, \boldsymbol{k}) \in \mathcal{B}$, and do so by showing it is maximized when $\boldsymbol{k} = \boldsymbol{0}$ and $m = 2|J| - 1$.

First, from Lemma 9, we have that for a given $(m, \boldsymbol{k}) \in \mathcal{B}$, we can find a pair in $\mathcal{B}$ of the form $(\tilde{m}, \boldsymbol{0})$ such that $G(\tilde{m}, s, J, \boldsymbol{0}, \boldsymbol{0}) \geq G(m, s, J, \boldsymbol{k}, \boldsymbol{0})$. Hence we can set $\boldsymbol{k} = \boldsymbol{0}$. Next, we use Lemma 8, which shows that for pairs in $\mathcal{B}$ of the form $(m, \boldsymbol{0})$, the function $G(m, s, J, \boldsymbol{k}, \boldsymbol{0})$ is maximized when $m = 2|J| - 1$.

Putting these two lemmas together, we get that for a given s and J, $G(m, s, J, \boldsymbol{k}, \boldsymbol{0}) \leq G(\tilde{m}, s, J, \boldsymbol{0}, \boldsymbol{0}) \leq G(2|J| - 1, s, J, \boldsymbol{0}, \boldsymbol{0}) = 2^{|J|} - 1$ for all $(m, \boldsymbol{k}) \in \mathcal{B}$. Hence

$$\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) \leq \frac{1}{2^{|J|}} \frac{b^m}{b^m - 1} (2^{|J|} - 1) = \frac{b^m}{b^m - 1} \frac{2^{|J|} - 1}{2^{|J|}} \leq 1,$$

which holds since $2^{|J|} \leq b^m$, as $b \geq 2$, and $m \geq |J|$. □

Having examined all possible cases, we can now state our main result.

Theorem 4 *If $\tilde{P}_n$ is a scrambled $(0, m, s)$−net in base b, then $H_n(A) \leq \mathrm{Vol}(A \times A)$ for any unanchored box $A \in \mathcal{A}$, and thus its pairwise sampling dependence index satisfies $\mathcal{E}_n(\tilde{P}_n) \leq 0$.*

Proof Using Theorem 3, we need to show that $\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) \leq 1$ for all $\boldsymbol{k}, \boldsymbol{d}, J$ for a $(0, m, s)$−net P_n, i.e., that condition (13) holds for a $(0, m, s)$−net. First, from Proposition 1, if $J \neq \emptyset$ and $m \geq |\boldsymbol{k}| + |\boldsymbol{d}|_J + 2|J|$ then

$$\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) \leq \frac{b^m}{b^m - 1} \left(\frac{2^{|J|} - b^{|\boldsymbol{k}| + |\boldsymbol{d}|_J + 2|J| - m}}{2^{|J|}} \right) \leq 1.$$

Next, from Proposition 2 we have that if $J \neq \emptyset$ and $m < |J|$ then

$$\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) \leq \frac{b^m}{b^m - 1} \frac{1}{2^{|J|}} \frac{b - 1}{b} \leq 1.$$

Then, using Proposition 3 we get that if $0 < |J| < m < |\boldsymbol{k}| + |\boldsymbol{d}|_J + 2|J|$ then

$$\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) \leq \frac{b^m}{b^m - 1} \frac{2^{|J|} - 1}{2^{|J|}} \leq 1.$$

Finally, if $J = \emptyset$ then

$$\begin{aligned}\tilde{m}_b(\boldsymbol{k}, \boldsymbol{d}, J; P_n) &= \frac{P((\mathbf{U}, \mathbf{V}) \in D(\boldsymbol{k}, \boldsymbol{d}, J))}{\mathrm{Vol}(D(\boldsymbol{k}, \boldsymbol{d}, J))} = \frac{1}{b^{-2|\boldsymbol{k}|}} b^{-|\boldsymbol{k}|} \frac{\max(b^{m-|\boldsymbol{k}|} - 1, 0)}{b^m - 1} \\ &= b^{|\boldsymbol{k}|} \frac{\max(b^{m-|\boldsymbol{k}|} - 1, 0)}{b^m - 1} = C_b(\boldsymbol{k}; P_n),\end{aligned}$$

which was shown to be smaller or equal to 1 in [10] for a $(0, m, s)-$net. □

Using Theorem 4, we obtain the following result, which shows that a scrambled net integrates the indicator function 1_A of any unanchored box A with variance no larger than the Monte Carlo estimator variance. To our knowledge, this was not previously known. What is well known is that since A is an axis-parallel box, then 1_A has bounded variation in the sense of Hardy and Krause and therefore a scrambled net has variance in $O(n^{-2}(\log n)^s)$ [9].

Proposition 4 *Let A be an unanchored box in $[0, 1)^s$. Let $\hat{\mu}_{n,A}$ be the estimator for $\mu_A = \mathrm{E}(1_A) = \mathrm{Vol}(A)$ based on a scrambled $(0, m, s)-$net in base b with $n = b^m$. Then $\mathrm{Var}(\hat{\mu}_{n,A}) \leq \mu_A(1 - \mu_A)/n$.*

Proof The result follows from the fact that $\mathrm{Var}(\hat{\mu}_{n,A}) = \mu_A(1 - \mu_A)/n + (H_n(A) - \mu_A^2)(n - 1)/n$, and then applying Theorem 4 to show that $(H_n(A) - \mu_A^2) \leq 0$. □

6 The Scrambling Advantage

We now give an example showing the advantage of scrambling over a digital shift, which is a simpler randomization. It uses a point set P_n with $C_b(\boldsymbol{k}; P_n) \leq 1$ such that $P((\mathbf{U}, \mathbf{V}) \in A \times A) > \mathrm{Vol}(\mathrm{A} \times \mathrm{A})$ for an anchored box A, for $(\mathbf{U}, \mathbf{V})$ a pair of distinct points randomly chosen from the digitally shifted point set $\tilde{P}_n^{dig}$. So even the less restrictive condition $\mathcal{E}_{n,0}(\tilde{P}_n^{dig}) \leq 0$ is not met. On the other hand, since $C_b(\boldsymbol{k}; P_n) \leq 1$, Theorem 4.16 in [10] implies that $P(\mathbf{U} \in A, \mathbf{V} \in A) \leq \mathrm{Vol}^2(A)$ for $(\mathbf{U}, \mathbf{V})$ randomly chosen from the scrambled point set $\tilde{P}_n$.

Example Consider the two-dimensional point set $P_n = \{(i/5, i/5), (i/5, ((i + 1) \bmod 5)/5), i = 0, \ldots, 4\}$. We first verify that $C_b(\boldsymbol{k}; P_n) \leq 1$: this clearly holds for $\boldsymbol{k} = \mathbf{0}$. For $\boldsymbol{k} \in \{(1, 0), (0, 1)\}$, we have $C_b(\boldsymbol{k}; P_n) = 5 \times 1/9$. And for $\boldsymbol{k}$ with $|\boldsymbol{k}| \geq 2$ we have $C_b(\boldsymbol{k}; P_n) = 0$. Now consider the box $A = [0, 1/10) \times [0, 2/5)$.

Let us compute $P((\mathbf{U}, \mathbf{V}) \in A \times A)$, where $(\mathbf{U}, \mathbf{V})$ is a pair of distinct points randomly chosen from $\tilde{P}_n^{dig}$, where $\tilde{P}_n^{dig} = P_n + \mathbf{v}$, where the addition is done digitwise and $\mathbf{v} \sim U(0,1)^2$. Let $\mathbf{v} = (v_1, v_2)$ with $v_j = 0.v_{j,1}v_{j,2}\ldots$, $j = 1, 2$. Then we see that among the 5^2 possibilities for $(v_{1,1}, v_{2,1})$, one point from $\tilde{P}_n^{dig}$ will be in the square $[0, 1/5) \times [0, 1/5)$ and one in the square $[0, 1/5) \times [1/5, 2/5)$ if and only if $v_{1,1} = v_{2,1}$, which happens with probability 1/5. Given that this happens, then it should also be clear that both points in that pair will be in A if and only if $(0.0v_{1,2}v_{1,3}\ldots, 0.0v_{2,2}v_{2,3}\ldots) \in [0, 1/10) \times [0, 1/5)$, which happens with probability 1/2 since $\mathbf{v} \sim U(0,1)^2$. Putting this all together, we get $P((\mathbf{U}, \mathbf{V}) \in A \times A) = \frac{1}{5}\frac{1}{45}\frac{1}{2} = \frac{1}{450}$, where the fraction $1/45$ corresponds to the probability of choosing the pair of points falling in the squares (0,0) and (0,1) among the 45 different (unordered) pairs. Since $\mathrm{Vol}(A) = 1/25$, we have that $P((\mathbf{U}, \mathbf{V}) \in A \times A) > (\mathrm{Vol}(A))^2 = 1/625$.

7 Future Work

In this paper, we have introduced a measure of uniformity for randomized QMC point sets that compares them to random sampling. This *pairwise sampling dependence index* was shown to be no larger than 0 for scrambled $(0, m, s)-$nets, thus extending from anchored boxes to unanchored boxes the main result from [10]. For future work, we plan to try to extend our proof to the first n points of a scrambled $(0, s)-$sequence. We also plan to explore how this result can lead to new bounds for the variance of scrambled $(0, m, s)-$nets in terms of the Monte Carlo variance for some functions.

Acknowledgements We thank the anonymous reviewers for their detailed comments. The first author thanks NSERC for their support via grant #238959. The second author wishes to acknowledge the support of the Austrian Science Fund (FWF): Projects F5506-N26 and F5509-N26, which are parts of the Special Research Program "Quasi-Monte Carlo Methods: Theory and Applications".

Appendix: Proofs and Technical Lemmas

We first prove results stated in the main part of the paper. These proofs make use of Lemmas 10 to 15, which are presented in the second part of the appendix.

Proof (of Lemma 2) In what follows, we will use the notation x_ℓ to represent the ℓth digit in the base b representation of $x \in [0, 1)$, i.e., $x = \sum_{\ell \geq 1} x_\ell b^{-\ell}$, and the corresponding notation $x = 0.x_1x_2x_3\ldots$.

First, we decompose A into three parts as $A_1 = [hb^{-r+1} + gb^{-r} - z, hb^{-r+1} + gb^{-r})$, $A_2 = [hb^{-r+1} + Gb^{-r}, hb^{-r+1} + Gb^{-r} + Z)$, $A_3 = [hb^{-r+1} + gb^{-r}, hb^{-r+1} + Gb^{-r})$. Hence we have

$$V_i(A \times A) = \sum_{\ell=1}^{3} V_i(A_\ell \times A_\ell) + 2(V_i(A_1 \times A_2) + V_i(A_1 \times A_3) + V_i(A_2 \times A_3)), \qquad i \geq 0. \quad (21)$$

Since A_1 and A_2 are both completely contained in the respective intervals $[hb^{-r+1} + (g-1)b^{-r}, hb^{-r+1} + gb^{-r})$ and $[hb^{-r+1} + Gb^{-r}, hb^{-r+1} + (G+1)b^{-r})$, any x in A_1 is of the form $0.h_1 \ldots h_{r-1}(g-1)x_{r+1}x_{r+2}\ldots$. Similarly, $y \in A_2$ is of the form $0.h_1 \ldots h_{r-1}(G)y_{r+1}y_{r+2}\ldots$. On the other hand, for $z \in A_3$ we have that $z_i = h_i$ for $i \leq r-1$, $z_r \in \{g, \ldots, G-1\}$, and $z_\ell \geq 0$ for $\ell > r$. From this we infer:

1. No pair of points from A_1 or A_2 can have less than r initial common digits, thus $V_i(A_\ell \times A_\ell) = 0$ for $i = 0, \ldots, r-1$ and $\ell = 1, 2$.
2. No pair of points from A_3 can have less than $r-1$ initial common digits, thus $V_i(A_3 \times A_3) = 0$ for $i = 0, \ldots, r-2$.
3. A pair of points from any two of the following subsets: A_1, A_2, $[hb^{-r+1} + \beta b^{-r}, hb^{-r+1} + (\beta+1)b^{-r}) \subseteq A_3$, where $\beta = g, \ldots, G-1$ has exactly $r-1$ initial common digits, thus $V_{r-1}(A_j \times A_\ell) = \mathrm{Vol}(A_j)\,\mathrm{Vol}(A_\ell)$ for $j \neq \ell$, $V_{r-1}(A_3 \times A_3) = (G-g)(G-g-1)b^{-2r}$ and $V_i(A_j \times A_\ell) = 0$ for $i \geq r$, $j \neq \ell$.

Note that this implies that $\tilde{V}_r(A_i \times A_i) = \mathrm{Vol}^2(A_i)$ for $i = 1, 2$, and (using item (3)) $\tilde{V}_r(A_3 \times A_3) = \mathrm{Vol}^2(A_3) - V_{r-1}(A_3 \times A_3) = (G-g)b^{-2r}$.

The above statements also allow us to simplify (21) as follows:

$$\begin{aligned} V_{r-1}(A \times A) &= V_{r-1}(A_3 \times A_3) + 2 \sum_{1 \leq i < \ell \leq 3} V_{r-1}(A_i \times A_\ell) \\ &= V_{r-1}(A_3 \times A_3) + 2 \sum_{1 \leq i < \ell \leq 3} \mathrm{Vol}(A_i)\mathrm{Vol}(A_\ell) \qquad (22) \end{aligned}$$

$$V_i(A \times A) = \sum_{\ell=1}^{3} V_i(A_\ell \times A_\ell) \quad i \geq r. \qquad (23)$$

To prove (ii), consider the mappings $\varphi_j : [0, 1) \to [0, 1)$, $1 \leq j \leq 3$ defined as:

$$\begin{aligned} \varphi_1(hb^{-r+1} + gb^{-r} - x) &= 1 - x, \qquad 0 \leq x < b^{-r} \\ \varphi_2(hb^{-r+1} + Gb^{-r} + x) &= x, \qquad 0 \leq x < b^{-r} \\ \varphi_3(hb^{-r+1} + gb^{-r} + x) &= x, \qquad 0 \leq x < (G-g)b^{-r}. \end{aligned}$$

All three are isometric mappings and such that $\varphi_1(A_1) = [1-z, 1)$, $\varphi_2(A_2) = [0, Z)$, and $\varphi_3(A_3) = [0, (G-g)b^{-r})$. Also, since φ_j simply amounts to changing the first r digits of a point in A_j (and applies the same change to all points in A_j), it implies

$$\gamma_b(\varphi_j(v_{j,\ell}), \varphi_j(v_{j,h})) = \gamma_b(v_{j,\ell}, v_{j,h}), \qquad j = 1, 2, 3,$$

where $v_{1,\ell} = hb^{-r+1} + gb^{-r} - x$, $v_{1,h} = hb^{-r+1} + gb^{-r} - y$, $v_{2,\ell} = hb^{-r+1} + Gb^{-r} + x$, $v_{2,h} = hb^{-r+1} + Gb^{-r} + y$, and $v_{3,\ell} = hb^{-r+1} + gb^{-r} + w$, $v_{3,h} = hb^{-r+1} + gb^{-r} + z$. Therefore

$$V_i(A_1 \times A_1) = V_i(\varphi_1(A_1) \times \varphi_1(A_1)) = V_i([1-z, 1) \times [1-z, 1)) = V_i([0, z), [0, z))$$
$$V_i(A_2 \times A_2) = V_i(\varphi_2(A_2) \times \varphi_2(A_2)) = V_i([0, Z) \times [0, Z))$$
$$V_i(A_3 \times A_3) = V_i(\varphi_3(A_3) \times \varphi_2(A_3)) = V_i([0, (G-g)b^{-r}) \times [0, (G-g)b^{-r})).$$

These intervals, being anchored at the origin, satisfy the assumptions of Lemma 2.6 from [10], which implies $bV_{i+1}(A_j \times A_j) - V_i(A_j \times A_j) \geq 0$ for $j = 1, 2, 3$ and $i \geq 0$. Combining this with (23), property (ii) in the statement of Lemma 2 is established.

For (iii), let us first assume $g = G$, and thus $A_3 = \emptyset$. Then, using (22), we get $V_{r-1}(A \times A) = 2\text{Vol}(A_1 \times A_2)$. Furthermore, $\tilde{V}_r(A, A) = \tilde{V}_r(A_1 \times A_1) + \tilde{V}_r(A_2 \times A_2) = \text{Vol}^2(A_1) + \text{Vol}^2(A_2)$. Since $2\text{Vol}(A_1 \times A_2) \leq \text{Vol}^2(A_1) + \text{Vol}^2(A_2)$, (iii) is proved.

Now assume $g < G$. In this case, we need to further refine A_1 and A_2 as:

$$A_1 = [hb^{-r+1} + gb^{-r} - db^{-(r+1)} - f, hb^{-r+1} + gb^{-r})$$
$$A_2 = [hb^{-r+1} + Gb^{-r}, hb^{-r+1} + Gb^{-r} + Db^{-(r+1)} + F),$$

where $0 \leq d, D \leq b-1$, $f, F \in [0, b^{-(r+1)})$. Using (22) and (23), we then write

$$\begin{aligned} V_{r-1}(A \times A) &= V_{r-1}(A_3 \times A_3) + 2 \sum_{1 \leq i < \ell \leq 3} \text{Vol}(A_i)\text{Vol}(A_\ell) \\ &= \frac{(G-g)(G-g-1)}{b^{2r}} + 2\left(\left(\frac{d}{b^{r+1}} + f\right)\left(\frac{D}{b^{r+1}} + F\right)\right. \\ &\quad \left. + \frac{G-g}{b^r}\left(\frac{d}{b^{r+1}} + f\right) + \frac{G-g}{b^r}\left(\frac{D}{b^{r+1}} + F\right)\right) \end{aligned}$$

$$\begin{aligned} V_r(A \times A) &= \sum_{\ell=1}^{3} V_r(A_\ell \times A_\ell) = V_r(A_1 \times A_1) + V_r(A_2 \times A_2) + \frac{G-g}{b^{2r}}\frac{b-1}{b} \\ &= \frac{(d-1)d}{b^{2(r+1)}} + \frac{2fd}{b^{r+1}} + \frac{(D-1)D}{b^{2(r+1)}} + \frac{2FD}{b^{r+1}} + \frac{G-g}{b^{2r}}\frac{b-1}{b} \\ \tilde{V}_r(A \times A) &= \left(\frac{d}{b^{r+1}} + f\right)^2 + \left(\frac{D}{b^{r+1}} + F\right)^2 + \frac{G-g}{b^{2r}}. \end{aligned}$$

The last equality for $V_r(A \times A)$ is obtained by observing that for (x, y) to be in $V_r(A_1 \times A_1)$, either (i) $x = 0.h_1 \ldots h_{r-1}(g-1)d_1 \ldots$ and $y = 0.h_1 \ldots h_{r-1}(g-1)d_2 \ldots$ with $d_1 \neq d_2 \in \{0, \ldots, d-1\}$, or (ii) one of them is of the form $z_1 + (0.h_1 \ldots h_{r-1}(g-1)d)$ with $z_1 \in [0, f)$ and the other is of the form

$0.h_1 \dots h_{r-1}(g-1)d_1 \dots$ with $d_1 \in \{0, \dots, d-1\}$. Case (i) contributes a volume of size $(d-1)db^{-2(r+1)}$ and case (ii) contributes $2fdb^{-(r+1)}$. A similar argument can be used to derive $V_r(A_2 \times A_2)$.

Therefore $V_{r-1}(A \times A) - \frac{b(b-2)}{b-1} V_r(A \times A) \leq \tilde{V}_r(A \times A)$ holds if

$$\begin{aligned}
&\frac{(G-g)(G-g-1)}{b^{2r}} + 2\left(\frac{G-g}{b^r}\left(\frac{d+D}{b^{r+1}} + f + F\right)\right) + 2\left(\frac{d}{b^{r+1}} + f\right)\left(\frac{D}{b^{r+1}} + F\right) \\
&- \frac{b(b-2)}{b-1}\left(\frac{(d-1)d}{b^{2(r+1)}} + \frac{2fd}{b^{r+1}} + \frac{(D-1)D}{b^{2(r+1)}} + \frac{2FD}{b^{r+1}} + \frac{G-g}{b^{2r}}\frac{b-1}{b}\right) \\
&\leq \left(\frac{d}{b^{r+1}} + f\right)^2 + \left(\frac{D}{b^{r+1}} + F\right)^2 + \frac{G-g}{b^{2r}}. \qquad (24)
\end{aligned}$$

Since

$$2\left(\frac{d}{b^{r+1}} + f\right)\left(\frac{D}{b^{r+1}} + F\right) \leq \left(\frac{d}{b^{r+1}} + f\right)^2 + \left(\frac{D}{b^{r+1}} + F\right)^2$$

it means that to prove (24) it is sufficient to show that

$$\begin{aligned}
&\frac{(G-g)(G-g-1)}{b^{2r}} - (b-2)\frac{G-g}{b^{2r}} + 2\left(\frac{G-g}{b^r}\left(\frac{d+D}{b^{r+1}} + f + F\right)\right) \\
&- \frac{b(b-2)}{b-1}\left(\frac{(d-1)d}{b^{2(r+1)}} + \frac{2fd}{b^{r+1}} + \frac{(D-1)D}{b^{2(r+1)}} + \frac{2FD}{b^{r+1}}\right) \leq \frac{G-g}{b^{2r}},
\end{aligned}$$

or equivalently, that

$$\begin{aligned}
&-b^2(G-g)(b-(G-g)) + 2b(G-g)\left(d + D + b^{r+1}(f+F)\right) \\
&- \frac{b(b-2)}{b-1}\left((d-1)d + 2fdb^{r+1} + (D-1)D + 2FDb^{r+1}\right) \leq 0. \qquad (25)
\end{aligned}$$

Note that $G - g \leq b - 2$ by assumption. We proceed by considering three cases:
Case 1: $G - g \leq b - 4$. This implies $b - (G - g) \geq 4$ (and thus $b \geq 4$). Also, to handle this case we use the fact that $0 \leq fb^{r+1}, Fb^{r+1} < 1$. By making appropriate substitutions for f and F, we see that to prove (25) holds it is sufficient to show that

$$-4b^2(G-g) + 2b(G-g)(d+D+2) - \frac{b(b-2)}{b-1}(d(d-1) + D(D-1)) \leq 0$$

which holds because $d + D + 2 \leq 2b$.
Case 2: $G - g = b - 3$ First note that this implies $b \geq 3$. Next, we replace $G - g$ with $b - 3$ in (25) and divide each term by b. For this case, we can use the bound $0 \leq fb^{r+1}, Fb^{r+1} < 1$ and by substituting appropriately, it means it is sufficient to show

$$-3b(b-3)+2(b-3)(d+D+2)-\frac{b-2}{b-1}(d(d-1)+D(D-1))\le 0.$$

We view the LHS as the sum of two quadratic polynomials, $p(d)$ and $p(D)$, and thus argue it is sufficient to show that

$$p(d):=\frac{-(b-2)}{b-1}d^2+d\left(2(b-3)+\frac{b-2}{b-1}\right)-(b-3)(3b/2-2)\le 0.$$

We will show this holds by finding the value d_{max} of d that maximizes $p(d)$ and show that $p(d_{max})\le 0$. We have that

$$p'(d)=-2d\frac{b-2}{b-1}+2(b-3)+\frac{b-2}{b-1}.$$

Therefore

$$d_{max}=\left(2(b-3)+\frac{b-2}{b-1}\right)\frac{b-1}{2(b-2)}=\frac{(b-3)(b-1)}{b-2}+\frac{1}{2}.$$

Hence $d_{max}\in(b-2.5, b-1.5)$. Thus it is sufficient to show $p(b-2)\le 0$. Now,

$$p(b-2)=-\frac{(b-2)^3}{b-1}+(b-2)\left(2(b-3)+\frac{b-2}{b-1}\right)-(b-3)(3b/2-2)$$

therefore

$$\begin{aligned}(b-1)p(b-2)&=-(b-2)^3+2(b-1)(b-2)(b-3)+(b-2)^2\\&\quad-\left(\frac{3b}{2}-2\right)(b-3)(b-1)\\&=(3-b)(b^2-3b+4)/2=(3-b)(b(b-3)+4)/2\le 0\end{aligned}$$

since $b\ge 3$.

Case 3: $G-g=b-2$

In this case (25) becomes

$$\begin{aligned}&-2b^2(b-2)+2b(b-2)(d+D+(f+F)b^{r+1})\\&-\frac{b(b-2)}{b-1}\left(d(d-1)+D(D-1)+2b^{r+1}(fd+FD)\right)\le 0\\\Leftrightarrow\ &-2b+2(d+D+(f+F)b^{r+1})\\&-\frac{1}{b-1}\left(d(d-1)+D(D-1)+2b^{r+1}(fd+FD)\right)\le 0.\end{aligned}$$

As in the case $G - g = b - 3$, we argue it is sufficient to show each of the quadratic polynomials in d and D on the LHS (which are the same) is bounded from above by 0. That is, we need to show

$$q(d) := \frac{-d^2}{b-1} + d\left(2 + \frac{1}{b-1} - \frac{2fb^{r+1}}{b-1}\right) - (b - 2fb^{r+1}) \leq 0.$$

Now

$$q'(d) = \frac{-2d}{b-1} + 2 + \frac{1}{b-1} - \frac{2fb^{r+1}}{b-1}$$

and thus $d_{max} = \left(2 + \frac{1}{b-1} - \frac{2fb^{r+1}}{b-1}\right)\frac{b-1}{2} = b - 0.5 - fb^{r+1}$, which implies $d_{max} \in (b - 1.5, b - 0.5)$. Thus it is sufficient to show $q(b-1) \leq 0$. We have that

$$\begin{aligned} q(b-1) &= -(b-1) + (b-1)\left(2 + \frac{1}{b-1} - \frac{2fb^{r+1}}{b-1}\right) - (b - 2fb^{r+1}) \\ &= (b-1) + 1 - 2fb^{r+1} - b + 2fb^{r+1} = 0 \end{aligned}$$

as required. □

Proof (of Theorem 1) To simplify the notation, we define $Z_k := b^{2k}V(1_k \times I_k)$ and $(W_k^{(r-1)} := (b^{2(k+r+1)}/4)V(Y_k^{(r-1)} \times Y_k^{(r-1)})$ to be the (normalized) volume vectors of $k-$elementary intervals and elementary unanchored $(k, r-1)-$intervals, respectively. Since the coordinates of each of these vectors are positive and sum to one, $V(A \times A) = \sum_{k\geq 0}(\alpha_k + \tau_k)$ follows immediately from the last equality in the statement of the theorem.

Based on the definition of $Y_k^{(r-1)}$, the vectors $W_k^{(r-1)}$ satisfy, for $i \geq 0, r \geq 1$,

$$W_{i,k}^{(r-1)} = \begin{cases} 1/2 & \text{if } i = r-1 \\ (b-1)/2b^{i-(k+r)} & \text{if } i \geq k+r+1 \\ 0 & \text{otherwise,} \end{cases}$$

while

$$Z_{i,k} = \begin{cases} 0 & \text{if } i < k \\ (b-1)/b^{i-k+1} & \text{if } i \geq k. \end{cases}$$

Note that

$$W_{r-1,k}^{(r-1)} = 1/2 \text{ for all } k \geq 0 \tag{26}$$

$$W_{i,k}^{(r-1)} = Z_{i,k+r+1}/2 \text{ for } i \geq k+r+1. \tag{27}$$

There are two cases to consider. First, if $V_{r-1}(A \times A) \leq bV_r(A \times A)$, then based on Properties (i) and (ii) from Lemma 2 we can decompose $V(A \times A)$ solely with the Z_k's , i.e., we set $\alpha_k = 0$ for all k and

$$\tau_k = \frac{bV_k(A \times A) - V_{k-1}(A \times A)}{b-1} \quad k \geq r-1$$

and $\tau_k = 0$ if $0 \leq k \leq r-2$. Note that $A = [0, 1)$ fits into this first case.

Second, if $V_{r-1}(A \times A) \geq bV_r(A \times A)$, then we first decompose the vector $\sum_{k=r}^{\infty} V_k(A \times A)\mathbf{e}_k$, where $\mathbf{e}_k$ is a (canonical) vector of zeros with a 1 in position k, (note that this agrees with the vector $V(A \times A)$ everywhere except on index $r-1$, where it has a 0 instead of $V_{r-1}(A \times A)$) as

$$\sum_{k=r}^{\infty} V_k(A \times A)\mathbf{e}_k = \sum_{k=r}^{\infty} \bar{\tau}_k Z_k$$

with $\bar{\tau}_r = bV_r(A \times A)/(b-1) \geq 0$, $\bar{\tau}_k = 0$ if $k < r$ (from Part (i) of Lemma 2), and

$$\bar{\tau}_k = \frac{bV_k(A \times A) - V_{k-1}(A \times A)}{b-1} \quad k \geq r+1.$$

From Lemma 2 we know $\bar{\tau}_k \geq 0$ for $k \geq r+1$. Note that $b\bar{\tau}_r Z_{r-1} - \bar{\tau}_r Z_r = bV_r(A \times A)\mathbf{e}_{r-1}$, i.e., $b\bar{\tau}_r Z_{r-1}$ agrees with $\bar{\tau}_r Z_r$ everywhere except on index $r-1$. Therefore

$$V(A \times A) - b\bar{\tau}_r Z_{r-1} - \sum_{k=r+1}^{\infty} \bar{\tau}_k Z_k = (V_{r-1}(A \times A) - bV_r(A \times A))\mathbf{e}_{r-1}.$$

Hence the $\sum_{k=0}^{\infty} \alpha_k W_k^{(r-1)}$ part of the decomposition is only needed to decompose $V_{r-1}(A \times A) - bV_r(A \times A)$. We claim there exists $\bar{\alpha}_k \geq 0$, $k \geq r+1$ such that

$$V_{r-1}(A \times A) - bV_r(A \times A) = \sum_{k=r+1}^{\infty} \bar{\alpha}_k/2, \tag{28}$$

and such that $\bar{\alpha}_k/2 \leq \bar{\tau}_k$ for $k \geq r+1$. This can be seen using Part (iii) of Lemma 2. Indeed, to ensure the existence of these $\bar{\alpha}_k$'s, we need to prove that

$$V_{r-1}(A \times A) - bV_r(A \times A) \leq \sum_{k=r+1} \bar{\tau}_k. \tag{29}$$

Now,

$$\sum_{k=r+1} \bar{\tau}_k = \sum_{k=r+1}^{\infty} \frac{bV_k(A\times A) - V_{k-1}(A\times A)}{b-1}$$
$$= \tilde{V}_{r+1}(A\times A) - \frac{V_r(A\times A)}{b-1} = \tilde{V}_r(A\times A) - V_r(A\times A) - \frac{V_r(A\times A)}{b-1}.$$

Therefore, (29) holds if and only if

$$V_{r-1}(A\times A) - bV_r(A\times A) \le \tilde{V}_r(A\times A) - V_r(A\times A) - \frac{V_r(A\times A)}{b-1}$$

which holds if and only if $V_{r-1}(A\times A) \le \tilde{V}_r(A\times A) + \frac{b(b-2)}{b-1} V_r(A\times A)$, which is precisely what Part (iii) of Lemma 2 shows. Having proved the existence of non-negative coefficients $\bar{\alpha}_k$ satisfying (28) implies we can write $V_{r-1}(A\times A) - bV_r(A\times A) = \sum_{k=r+1}^{\infty} \bar{\alpha}_k W_{r-1,k-(r+1)}^{(r-1)}$ by using (26). Hence all that is left to do is to find the combination of Z_k's that can cancel out $\sum_{k=r+1}^{\infty} \bar{\alpha}_k W_{i,k-(r+1)}^{(r-1)}$ for $i \ge r+1$ (we can ignore the case $i = r$ because $W_{r,k-(r+1)}^{(r-1)} = 0$ for all $k \ge r+1$). This is done by using (27), which implies that

$$\sum_{k=r+1}^{\infty} \bar{\alpha}_k W_{i,k-(r+1)}^{(r-1)} = \sum_{k=r+1}^{\infty} (\bar{\alpha}_k/2) Z_{i,k}.$$

Hence the final decomposition is given by

$$V(A\times A) = \sum_{k=r+1}^{\infty} \bar{\alpha}_k W_{k-(r+1)}^{(r-1)} + b\bar{\tau}_r Z_{r-1} + \sum_{k=r+1}^{\infty} (\bar{\tau}_k - \bar{\alpha}_k/2) Z_k$$
$$= \sum_{k=0}^{\infty} \alpha_k W_k^{(r-1)} + \sum_{k=0}^{\infty} \tau_k Z_k$$

with $\alpha_k = \bar{\alpha}_{k+(r+1)}$, $k \ge 0$, $\tau_k = \bar{\tau}_k - \bar{\alpha}_k/2$, $k \ge r+1$, $\tau_{r-1} = b\bar{\tau}_r$ and $\tau_k = 0$ for $0 \le k \le r-2, k = r$. □

Proof (of Lemma 3) (1) From the definition of $D(\boldsymbol{k}, \boldsymbol{d}, J)$, we have that

$$\mathrm{Vol}(D(\boldsymbol{k}, \boldsymbol{d}, J)) = b^{-2|\boldsymbol{k}|_{J^c}} 2^{2|J|} b^{-2(|\boldsymbol{k}+\boldsymbol{d}+2|_J)} = 2^{2|J|} b^{-2(|\boldsymbol{k}|+|\boldsymbol{d}+2|_J)}.$$

(2) Similarly, from the definition of $F_1(\boldsymbol{k}, \boldsymbol{d}, J, I)$ we get

$$\mathrm{Vol}(F_1(\boldsymbol{k}, \boldsymbol{d}, J, I)) = b^{-|\boldsymbol{k}|_{J^c}} b^{-(|\boldsymbol{k}+\boldsymbol{d}+2|_J)} = b^{-(|\boldsymbol{k}|+|\boldsymbol{d}+2|_J)}.$$

(3) This conditional probability is given by $\eta/n - 1$, where η is the number of points $\mathbf{u}_\ell$ with $\ell \ne i$ that are in F_2 if $\mathbf{u}_i \in F_1$. Hence for $j \in I$ we must have $\gamma_b(u_{i,j}, u_{\ell,j}) \ge$

$k_j + d_j + 2$; for $j \in I^c$ we must have $\gamma_b(u_{i,j}, u_{\ell,j}) \geq k_j$. For $j \in J \cap I^c$, the requirement that $u_{i,j} \in F_1$ means $u_{\ell,j}$ must satisfy:

(a) it must have the same first d_j digits as $u_{i,j}$;
(b) its $(d_j + 1)$th digit must be 1 (while the $(d_j + 1)$th digit of $u_{i,j}$ is 0);
(c) the digits $u_{\ell,j,r}$ for $d_j + 2 \leq r d_j + k_j + 2$ must be 0

If we only had to satisfy requirement (a), then we would have $\eta = m_b(\boldsymbol{k}, \boldsymbol{d}, 2, J, I)$. However, the requirements (b) and (c) imply

$$\eta = m_b(\boldsymbol{k}, \boldsymbol{d}, 2, J, I) \prod_{j \in J \cap I^c} \frac{1}{b-1} \frac{1}{b^{k_j+1}},$$

where the term $1/(b-1)$ handles restriction (b) while the term b^{-k_j-1} handles restriction (c). Therefore

$$\begin{aligned} P(\mathbf{V} \in F_2(\boldsymbol{k}, \boldsymbol{d}, J, I) | \mathbf{U} \in F_1(\boldsymbol{k}, \boldsymbol{d}, J, I)) &= \frac{m_b(\boldsymbol{k}, \boldsymbol{d}, 2, J, I)}{n-1} \left(\frac{1}{b-1}\right)^{|J|-|I|} \frac{1}{b^{|\boldsymbol{k}+1|_{J \cap I^c}}} \\ &= \frac{m_b(\boldsymbol{k}, \boldsymbol{d}, 2, J, I)}{n-1} \frac{(b-1)^{|I|-|J|}}{b^{|\boldsymbol{k}|_{J \cap I^c} + |J| - |I|}}. \end{aligned}$$

(4) The decomposition $\cup_{K, I \subseteq J} F(\boldsymbol{k}, \boldsymbol{d}, J, I, K)$ is obtained by expanding each $Y_{k_j}^{(d_j)}$ as $Y_{k_j,1}^{(d_j)} \cup Y_{k_j,2}^{(d_j)}$. Then, we need to prove that for $K_1, K_2 \subseteq J$,

$$P(\mathbf{U}, \mathbf{V}) \in F(\boldsymbol{k}, \boldsymbol{d}, J, I, K_1) = P(\mathbf{U}, \mathbf{V}) \in F(\boldsymbol{k}, \boldsymbol{d}, J, I, K_2). \tag{30}$$

Noting that the equality (7) can be generalized to $P((\mathbf{U}, \mathbf{V}) \in \mathcal{R}) = \sum_{\boldsymbol{i} \geq \boldsymbol{0}} \psi_{\boldsymbol{i}} V_{\boldsymbol{i}}(\mathcal{R})$, it is clear that to prove (30), it is sufficient to show that the volume vectors corresponding to $F(\boldsymbol{k}, \boldsymbol{d}, J, I, K_1)$ and $F(\boldsymbol{k}, \boldsymbol{d}, J, I, K_2)$ are equal. To do so, since each entry $V_{\boldsymbol{i}}(\mathcal{R}) = \prod_{j=1}^{s} V_{i_j}(\mathcal{R}_j)$ (where for $\mathcal{R} = \mathcal{R}_1 \times \mathcal{R}_2$ we write $\mathcal{R}_j = \mathcal{R}_{1,j} \times \mathcal{R}_{2,j}$), it is sufficient to show that for fixed k and d, $V_{11} := V(Y_{k,1}^{(d)} \times Y_{k,1}^{(d)}) = V(Y_{k,2}^{(d)} \times Y_{k,2}^{(d)}) =: V_{2,2}$ and $V_{12} := V(Y_{k,1}^{(d)} \times Y_{k,2}^{(d)}) = V(Y_{k,2}^{(d)} \times Y_{k,1}^{(d)}) =: V_{2,1}$. But this follows from an argument similar to the one used in the proof of Lemma 4.14 in [10], which we adapt for our setup. First we introduce the set $\mathcal{F} = \{ab^{-k} : a \in \mathbb{Z}, k \in \mathbb{N}\} \subseteq \mathbb{R}$ which has Lebesgue measure 0. Then, we argue that for $x, y \in (0, b^{-(k+d+2)}) \cap \mathcal{F}^c$, we have

$$\begin{aligned} \gamma_b\left(\frac{1}{b^{d+1}} - x, \frac{1}{b^{d+1}} - y\right) &= \gamma_b\left(\frac{1}{b^{d+1}} + x, \frac{1}{b^{d+1}} + y\right) \\ \gamma_b\left(\frac{1}{b^{d+1}} - x, \frac{1}{b^{d+1}} + y\right) &= \gamma_b\left(\frac{1}{b^{d+1}} + x, \frac{1}{b^{d+1}} - y\right). \end{aligned}$$

Therefore, for $(x, y) \in Y_k^{(d)} \times Y_k^{(d)}$, D_i is, up to a set of measure 0, invariant under the transformation $(x, y) \mapsto \left(\frac{2}{b^{d+1}} - x, \frac{2}{b^{d+1}} - y\right)$. This transformation maps $Y_{k,1}^{(d)} \times$

$Y_{k,1}^{(d)}$ to $Y_{k,2}^{(d)} \times Y_{k,2}^{(d)}$ (and vice-versa) and $Y_{k,1}^{(d)} \times Y_{k,2}^{(d)}$ to $Y_{k,2}^{(d)} \times Y_{k,1}^{(d)}$ (and vice-versa). Therefore $V_{11} = V_{22}$, and $V_{12} = V_{21}$, as required. □

Proof (of Lemma 4) First, if $\ell = 2i + 1$, then

$$g_{i,j}(\ell) = \left(\frac{b}{b-1}\right)^{j-i-1} \leq \left(\frac{b}{b-1}\right)^{i+j-\ell}$$

and

$$\left(\frac{b}{b-1}\right)^{i+j-\ell} = \left(\frac{b}{b-1}\right)^{i+1-j}$$

so in this case we actually have an equality, i.e.,

$$g_{i,j}(\ell) = \left(\frac{b}{b-1}\right)^{i+j-\ell}.$$

If $2i < \ell < i + j$ and ℓ is even, then

$$\left(\frac{b}{b-1}\right)^{i+j-\ell} \geq 1 = g_{j,i}(\ell)$$

since $\ell < i + j$. If $2i < \ell < i + j$ and ℓ is odd, then we must show that

$$1 + \frac{b^{j+i-\ell}}{(b-1)^{j-i}} \binom{j-i-1}{\ell-2i} \leq \left(\frac{b}{b-1}\right)^{i+j-\ell}. \tag{31}$$

Now let $k = \ell - 2i$ and $r = j - i$. This means k is odd and $k > 1$, and also $k < r \leq j \leq s$ which means $r \geq 4$. Using this notation, (31) is equivalent to

$$b^k \left(\frac{b-1}{b}\right)^r + \binom{r-1}{k} \leq (b-1)^k \text{ and thus to } \left(\frac{b-1}{b}\right)^{r-k} + \frac{\binom{r-1}{k}}{(b-1)^k} \leq 1. \tag{32}$$

Now,

$$\left(\frac{b-1}{b}\right)^{r-k} = \sum_{j=0}^{r-k} \binom{r}{j} \left(\frac{-1}{b}\right)^j$$

and $\binom{r}{j}/b^j$ is decreasing with j, therefore

$$\left(\frac{b-1}{b}\right)^{r-k} \leq 1 - \frac{r-k}{b} + \frac{(r-k)(r-k-1)}{2b^2},$$

because the condition that $k > 1$ and k is odd implies $k \geq 3$. Therefore a sufficient condition for the second inequality in (32) to hold is if we have

$$1 - \frac{r-k}{b} + \frac{(r-k)(r-k-1)}{2b^2} + \frac{(r-1)(r-2)\dots(r-k)}{(b-1)(b-1)\dots(b-1)}\frac{1}{k!} \le 1$$

which holds iff $$\frac{(r-k)(r-k-1)}{2b^2} + \frac{(r-1)(r-2)\dots(r-k)}{(b-1)(b-1)\dots(b-1)}\frac{1}{k!} \le \frac{r-k}{b}$$

which holds iff $$\frac{r-k-1}{2b} + \frac{(r-1)(r-2)\dots(r-k+1)}{(b-1)(b-1)\dots(b-1)}\frac{b}{b-1}\frac{1}{k!} \le 1. \tag{33}$$

Now,

$$\frac{r-k-1}{2b} < \frac{1}{2} \Leftrightarrow r-k-1 < b,$$

and the latter inequality holds since $b \ge s \ge r$ and $k \ge 3$. Also,

$$\frac{r-k+1}{2(b-1)} \le \frac{1}{2} \Leftrightarrow r-k+1 \le b-1 \Leftrightarrow r+2-k \le b$$

and the latter inequality holds since $k \ge 3$ and $b \ge s \ge r$. Therefore, for $k \ge 3$ the LHS of (33) is bounded (strictly) from above by

$$\frac{1}{2} + \frac{1}{2}\frac{(r-1)(r-2)\dots(r-k+2)}{(b-1)(b-1)\dots(b-1)}\frac{b}{b-1}\frac{1}{k(k-1)\dots 3} < 1$$

because

$$\frac{(r-1)(r-2)\dots(r-k+2)}{(b-1)(b-1)\dots(b-1)} \le 1$$

since $b \ge s \ge r$ and

$$\frac{b}{b-1}\frac{1}{k(k-1)\dots 3} \le \frac{4}{3} \times \frac{1}{3} < 1$$

since $b/(b-1)$ decreases with b, and the condition $1 < k < r \le s$ with $k \ge 3$ means we can assume $b \ge s \ge r \ge 4$. This proves that (32) holds. □

Proof (of Lemma 5) (i) Using Lemma 12 with $c = 2$ (which implies $c|I| + |I^{*c}| = |J| + |I|$), we first consider the case where $m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J \ge |I| + |J|$. In this case

$$m(\boldsymbol{k}, \boldsymbol{d}, 2, J, I) = (b-1)^{|J|-|I|} b^{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J-|J|-|I|}.$$

Hence if $m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J \ge |I| + |J|$, then

$$\begin{aligned}\psi_m(\boldsymbol{k}, \boldsymbol{d}, J, I) &= \frac{(b-1)^{|J|-|I|}}{n-1} b^{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J-|J|-|I|} b^{|\boldsymbol{k}|_{I^*}+|\boldsymbol{d}|_J+|J|+|I|} (b-1)^{|I|-|J|} \\ &= \frac{b^m}{b^m-1}.\end{aligned}$$

(ii) Next, again based on Lemma 12, we consider the case $2|I| + 1 < m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J < |I| + |J|$. First, if $m - |\boldsymbol{k}|_{I^c} - |\boldsymbol{d}|_J$ is odd and $m - |\boldsymbol{k}|_{I^c} - |\boldsymbol{d}|_J > 2|I| + 1$,

then

$$m(\boldsymbol{k},\boldsymbol{d},2,J,I) \le \left(b^{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J-2|I|}\left(\frac{b-1}{b}\right)^{|J|-|I|} + \binom{|J|-|I|-1}{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J-2|I|}\right). \tag{34}$$

Hence in that case

$$\begin{aligned}\psi_m(\boldsymbol{k},\boldsymbol{d},J,I) &\le \frac{1}{b^m-1} b^{|\boldsymbol{k}|_{I^*}+|\boldsymbol{d}|_J+|J|+|I|}(b-1)^{|I|-|J|} \\ &\quad\cdot\left(b^{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J-2|I|}\left(\frac{b-1}{b}\right)^{|J|-|I|} + \binom{|J|-|I|-1}{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J-2|I|}\right) \\ &= \frac{1}{b^m-1}\left(b^m + b^{|\boldsymbol{k}|_{I^*}+|\boldsymbol{d}|_J+|J|+|I|}(b-1)^{|I|-|J|}\binom{|J|-|I|-1}{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J-2|I|}\right) \\ &= \frac{b^m}{b^m-1}\left(1+\frac{b^{|J|+|I|-(m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J)}}{(b-1)^{|J|-|I|}}\binom{|J|-|I|-1}{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J-2|I|}\right).\end{aligned}$$

A similar calculation shows that if $m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J$ is even then

$$\psi_m(\boldsymbol{k},\boldsymbol{d},J,I) \le \frac{b^m}{b^m-1}.$$

If $m-|\boldsymbol{k}|_{I^c}-|\boldsymbol{d}|_J-2|I|=1$, then from Lemma 12 we know that $m(\boldsymbol{k},\boldsymbol{d},2,J,I)=(b-1)$, and thus

$$\begin{aligned}\psi_m(\boldsymbol{k},\boldsymbol{d},J,I) &= \frac{1}{b^m-1} b^{|\boldsymbol{k}|_{I^*}+|\boldsymbol{d}|_J+|J|+|I|}(b-1)^{|I|-|J|}(b-1) \\ &= \frac{b^m}{b^m-1} b^{|J|-|I|-1}(b-1)^{1+|I|-|J|} = \frac{b^m}{b^m-1}\left(\frac{b}{b-1}\right)^{|J|-|I|-1}.\end{aligned}$$

Combining these three cases, we get that for $2|I| < m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J < |J|+|I|$,

$$\psi_m(\boldsymbol{k},\boldsymbol{d},J,I) = \frac{b^m}{b^m-1} g_{|J|,|I|}(m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}_J)$$

with

$$g_{|J|,|I|}(m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}_J) = \begin{cases} 1+h_{|J|,|I|}(m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J) & \text{if } m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J > 2|I|+1 \\ & \text{and } m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J \text{ is odd} \\ \left(\frac{b}{b-1}\right)^{|J|-|I|-1} & \text{if } m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J = 2|I|+1 \\ 1 & \text{if } m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J \text{ is even,} \end{cases}$$

where

$$h_{j,i}(\ell) = \frac{b^{j+i-\ell}}{(b-1)^{j-i}} \binom{j-i-1}{\ell-2i}.$$

(iii) When $m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J \le 2|I|$, then $m(\boldsymbol{k}, \boldsymbol{d}, 2, J, I) = 0$ and therefore we can set $g_{|J|,|I|}(m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J) = 0$. □

Proof (of Lemma 7) First we observe that

$$R(b,s) = \left(\frac{b}{2(b-1)}\right)^s P_{\tilde{s},s}((b-1)/b),$$

where $P_{m,n}(z)$ is the polynomial defined in Lemma 10, and recall that $\tilde{s} = \lfloor s/2 \rfloor - 1$. In the notation of (41), $z = (b-1)/b$, $z/(z+1) = (b-1)/(2b-1)$, and $1 + z = (2b-1)/b$. Therefore

$$R(b,s) = \left(\frac{b}{2(b-1)}\right)^s \left(\frac{2b-1}{b}\right)^s Pr\left(X > \frac{b-1}{2b-1}\right),$$

where X is a Beta rv with parameters $\tilde{s}+1, s-\tilde{s}$. Now, it is known that a beta distribution with parameters a, c such that $1 < a < c$ has a median no larger than $a/(a+c)$. Therefore, if we can show that

$$\frac{\tilde{s}+1}{s+1} \le \frac{b-1}{2b-1},$$

then it means $Pr(X > (b-1)/(2b-1)) \le 1/2$. Now,

$$\frac{b-1}{2b-1} = \frac{1}{2} - \frac{1}{2(2b-1)} \ge \frac{1}{2} - \frac{1}{2(2s-1)}.$$

On the other hand,

$$\frac{\tilde{s}+1}{s+1} \le \frac{s}{2(s+1)} = \frac{1}{2} - \frac{1}{2(s+1)}.$$

Since $2(2s-1) \ge 2(s+1)$ for $s \ge 2$, we have that

$$\frac{\tilde{s}+1}{s+1} \le \frac{1}{2} - \frac{1}{2(s+1)} \le \frac{1}{2} - \frac{1}{2(2s-1)} \le \frac{b-1}{2b-1},$$

as required. So the last step is to show that

$$\frac{1}{2}\left(\frac{b}{2(b-1)}\right)^s \left(\frac{2b-1}{b}\right)^s \le 1. \tag{35}$$

The inequality (35) is equivalent to

$$\left(\frac{2b-1}{2b-2}\right)^s \leq 2, \text{ to } s\ln\frac{2b-1}{2b-2} \leq \ln 2, \text{ and finally to } s\ln(1+\frac{1}{2(b-1)}) \leq \ln 2.$$

Now, $\ln(1+\frac{1}{2(b-1)}) \leq \ln(1+\frac{1}{2(s-1)}) \leq \frac{1}{2(s-1)}$, hence it is sufficient to show that

$$\frac{1}{2(s-1)} \leq \frac{\ln 2}{s} \Leftrightarrow \frac{s}{s-1} \leq 2\ln 2 = 1.3862...$$

which holds for any $s \geq 4$. For $s = 2$, we have that $R(b,2) = 0.25(b/(b-1))^2$ and since $b \geq s$ we have that $b/(b-1) \leq 2$, which implies $0.25(b/(b-1))^2 \leq 1$ for all $b \geq 2$. When $s = 3$, then $R(b,3) = 0.125(b/(b-1))^3 \leq 1$. □

Proof (of Lemma 8) We have that $(m, \mathbf{0}) \in \mathcal{B}$ is equivalent to assuming $|J| \leq m < 2|J|$. We will deal with the case $m = 2|J|-1$ separately, and will first assume $|J| \leq m \leq 2|J|-3$ and m is odd.

(i) Case where $|J| \leq m \leq 2|J|-3$ and m is odd:

In this case, $m^* = m > 2|I|$ if and only if $|I| < 0.5m$, i.e., $|I| \leq 0.5(m-1)$. Also, $m^* = m < |I| + |J|$ if and only if $|I| > m - |J|$. So $G(m, s, J, \mathbf{0}, \mathbf{0})$ is of the form

$$\begin{aligned}
&G(m,s,J,\mathbf{0},\mathbf{0}) \\
&= \sum_{j=0}^{0.5(m-3)} \binom{|J|}{j} 1 + \sum_{j=m-|J|+1}^{0.5(m-3)} \binom{|J|}{j} h_j(m) + \binom{|J|}{0.5(m-1)} \left(\frac{b}{b-1}\right)^{|I|-0.5(m-1)-1} \\
&= \sum_{j=0}^{0.5(m-3)} \binom{|J|}{j} + \sum_{j=m-|J|+1}^{0.5(m-3)} \binom{|J|}{j}\binom{|J|-j-1}{m-2j} \frac{(b(b-1))^j}{(b-1)^{|J|} b^{m-|J|}} \\
&\quad + \binom{|J|}{0.5(m-1)} \left(\frac{b}{b-1}\right)^{|J|-0.5(m-1)-1}. \qquad (36)
\end{aligned}$$

Using Lemma 14 with $s = |J|$, we know that (36) is increasing with m, so $G(m,s,J,\mathbf{0},\mathbf{0}) \leq G(2|J|-3,s,J,\mathbf{0},\mathbf{0})$ for all $m \leq 2|J|-3$ such that $(m,\mathbf{0}) \in \mathcal{B}$. Furthermore, we have that

$$\begin{aligned}
G(2|J|-3,s,J,\mathbf{0},\mathbf{0}) &= \sum_{j=0}^{|J|-3} \binom{|J|}{j} + \binom{|J|}{2}\binom{1}{1}\frac{b}{(b-1)} \\
&= 2^{|J|} - 1 - |J| + \frac{|J|(|J|-1)}{2}\left(\frac{b}{b-1} - 1\right) \\
&= 2^{|J|} - 1 - |J| + \frac{|J|(|J|-1)}{2(b-1)} \leq 2^{|J|} - 1 - \frac{|J|}{2},
\end{aligned}$$

where the last inequality is obtained by observing that $|J| \leq s \leq b$.

(ii) Case where $m = 2|J|-1$.

In this case, $m \geq |I| + |J|$ for all I such that $|I| \leq |J| - 2$. Also, for subsets I such that $|I| = |J| - 1$, we cannot have $2|I| < m < |I| + |J|$ since in that case $|I| + |J| - 2|I| = |J| - |I| = 1$. Therefore, there is no I such that $2|I| < m < |I| + |J|$, and thus

$$G(2|J| - 1, s, J, \mathbf{0}, \mathbf{0}) = \sum_{j=0}^{|J|-1} \binom{|J|}{j} = 2^{|J|} - 1 \geq G(2|J| - 3, s, J, \mathbf{0}, \mathbf{0}). \quad (37)$$

(iii) If m is even with $|J| \leq m < 2|J|$, then $G(m, s, J, \mathbf{0}, \mathbf{0}) \leq \sum_{j=0}^{|J|-1} \binom{|J|}{j} = 2^{|J|} - 1$. □

Proof (of Lemma 9) We let $j = |J|$ and write

$$G(m, \boldsymbol{k}) = \sum_{I:|I|\leq j-1} g_{j,|I|}(m - |\boldsymbol{k}|_{I^*}).$$

That is, $G(m, \boldsymbol{k}) = G(m, s, J, \boldsymbol{k}, \mathbf{0})$, i.e., we drop the dependence on s, J and $\boldsymbol{d}$.

First, we define ι as the size of the largest (strict) subset I of J that contributes a non-zero value to $G(m, \boldsymbol{k})$. That is,

$$\iota := \max\{|I| : I \subset J, m - |\boldsymbol{k}|_{I^*} \geq 2|I| + 1\}.$$

Note that $0 \leq \iota \leq j - 1$. Also, it is useful at this point to mention that our optimal solution $(\tilde{m}, \mathbf{0})$ will be such that $\tilde{m} = 2\iota + 1$.

We then define $\mathcal{G}_{m,\boldsymbol{k}} := \{I : 0 \leq |I| \leq \iota\}$. Using this notation we can write

$$G(m, \boldsymbol{k}) = \sum_{I:I\in\mathcal{G}_{m,\boldsymbol{k}}} g_{j,|I|}(m - |\boldsymbol{k}|_{I^*}). \quad (38)$$

This holds because if $I \notin \mathcal{G}_{m,\boldsymbol{k}}$, then $m - |\boldsymbol{k}|_{I^*} \leq 2|I|$ and thus $g_{j,|I|}(m - |\boldsymbol{k}|_{I^*}) = 0$.

Next we introduce a definition:

Definition: For a given J and $I \subset J$, we say that $(m, \boldsymbol{k})$ *is dominated by* $(m', \mathbf{0})$ *at* I if $g_{j,|I|}(m - |\boldsymbol{k}|_{I^*}) \leq g_{j,|I|}(m')$.

Our strategy will be as follows: consider the set $\mathcal{M} = \{1, 3, \ldots, 2\iota + 1\}$. We claim that for each $I \in \mathcal{G}_{m,\boldsymbol{k}}$, there exists $(m', \mathbf{0})$ with $m' \in \mathcal{M}$ such that $(m, \boldsymbol{k})$ is dominated by $(m', \mathbf{0})$ at I. In turn, this will allow us to bound each term $g_{j,|I|}(m - |\boldsymbol{k}|_{I^*})$ in (38) by a term of the form $g_{j,|I|}(m')$. We then only need to keep track, for each m', of how many times $g_{j,|I|}(m')$ has been used in this way—something we will do by introducing counting numbers denoted by $\eta(\cdot)$. This strategy is a key intermediate step to get to our end result, which is to show that $G(m, \boldsymbol{k}) \leq G(\tilde{m}, \mathbf{0})$.

To prove the existence of this m', we define a mapping $\mathcal{L}_{m,\boldsymbol{k}} : \mathcal{G}_{m,\boldsymbol{k}} \to \mathcal{M}$ that will, for a given m and $\boldsymbol{k}$, assign to each subset $I \in \mathcal{G}_{m,\boldsymbol{k}}$, the largest integer $m' \in \mathcal{M}'$ such that $(m, \boldsymbol{k})$ is dominated by $(m', \mathbf{0})$ at I. The reason why we choose the largest

$m' \in \mathcal{M}$ is that this provides us with the tightest bound on $g_{j,|I|}(m - |\boldsymbol{k}|_{I^*})$, as should be clear from the behavior of the function $g_{j,i}(\ell)$, as described in Lemma 13.

The mapping $\mathcal{L}_{m,\boldsymbol{k}}(I)$ is defined as follows:

$$\mathcal{L}_{m,\boldsymbol{k}}(I) := \begin{cases} 2\iota + 1 & \text{if } m - |\boldsymbol{k}|_{I^*} \geq 2\iota + 1 \\ 2\ell + 1 & \text{if } 2\ell + 1 \leq m - |\boldsymbol{k}|_{I^*} \leq 2\ell + 2 \text{ where } 0 \leq \ell < \iota \\ 1 & \text{if } m - |\boldsymbol{k}|_{I^*} \leq 0. \end{cases}$$

Claim: For each $I \in \tilde{\mathcal{G}}_{m,\boldsymbol{k}}$, $(\mathcal{L}_{m,\boldsymbol{k}}(I), \boldsymbol{0})$ dominates $(m, \boldsymbol{k})$ at I.
Proof: We need to show that $g_{j,|I|}(m - |\boldsymbol{k}|_{I^*}) \leq g_{j,|I|}(\mathcal{L}_{m,\boldsymbol{k}}(I))$. We proceed by examining the three possible cases for $\mathcal{L}_{m,\boldsymbol{k}}(I)$ based on its definition.

(i) Assume $m - |\boldsymbol{k}|_{I^*} \geq 2\iota + 1$ and therefore $\mathcal{L}_{m,\boldsymbol{k}}(I) = 2\iota + 1$. By definition of ι we have $2\iota + 1 > 2i$, and therefore Part 2 of Lemma 13 applies, which implies that $g_{j,|I|}(2\iota + 1) \geq g_{j,|I|}(m - |\boldsymbol{k}|_{I^*})$. (ii) We now assume $2\ell + 1 < m - |\boldsymbol{k}|_{I^*} \leq 2\ell + 2$ for some $0 \leq \ell < \iota$. In this case, $\mathcal{L}_{m,\boldsymbol{k}}(I) = 2\ell + 1$ and $m - |\boldsymbol{k}|_{I^*}$ is either equal to $2\ell + 1$ or to $2\ell + 2$. If $m - |\boldsymbol{k}|_{I^*} = 2\ell + 1$ then clearly $g_{j,|I|}(\mathcal{L}_{m,\boldsymbol{k}}(I)) \geq g_{j,|I|}(m - |\boldsymbol{k}|_{I^*})$ since in fact these two quantities are equal. If $m - |\boldsymbol{k}|_{I^*} = 2\ell + 2$ then since $\mathcal{L}_{m,\boldsymbol{k}}(I) = 2\ell + 1 \geq 1$ is odd we can use Part 1 of Lemma 13 to conclude that $g_{j,|I|}(\mathcal{L}_{m,\boldsymbol{k}}(I)) \geq g_{j,|I|}(m - |\boldsymbol{k}|_{I^*})$. (iii) If $m - |\boldsymbol{k}|_{I^*} \leq 0$, then $m - |\boldsymbol{k}|_{I^*} \leq 2|I|$ since $I \in \mathcal{G}_{m,\boldsymbol{k}}$ implies $0 \leq |I| \leq \iota$, and therefore $g_{j,|I|}(m - |\boldsymbol{k}|_{I^*}) = 0 \leq g_{j,|I|}(1)$. □

Now, recall that shortly after stating (38), when we explained our strategy to replace the terms $g_{j,|I|}(m - |\boldsymbol{k}|_{I^*})$ by $g_{j,|I|}(m')$ in (38), we also said we would need counting numbers $\eta(\cdot)$ to tell us how many times, for each m', the term $g_{j,|I|}(m')$ has been used in this way. These counting numbers are essential to apply the optimization result involving weighted sums that is given in Lemma 15, which is the key to get our final result. They are defined as follows, for $0 \leq i, \ell \leq \iota$:

$$\eta(\ell, i, \boldsymbol{k}) := |\{I \in \mathcal{G}_{m,\boldsymbol{k}} : |I| = i, \mathcal{L}_{m,\boldsymbol{k}}(I) = 2(\iota - \ell) + 1\}|.$$

Note that $\sum_{\ell=0}^{\iota} \eta(\ell, i, \boldsymbol{k}) = \binom{j}{i}$. Also we can think of $p(\ell, i, \boldsymbol{k}) = \eta(\ell, i, \boldsymbol{k})/\binom{j}{i}$ as the probability that a randomly chosen subset I of i elements from J is such that $|\boldsymbol{k}|_{I^*} \in \mathcal{R}_\ell$, where

$$\mathcal{R}_\ell := \begin{cases} \{0, 1, \ldots, m - (2\iota + 1)\} & \text{if } \ell = 0 \\ \{m - (2\iota + 1) + 2\ell - 1, m - (2\iota + 1) + 2\ell)\} & \text{if } 1 \leq \ell < \iota \\ \{m - 2, m - 1, \ldots\} & \text{if } \ell = \iota. \end{cases} \tag{39}$$

To get the final result, we write:

$$
\begin{aligned}
G(m, \boldsymbol{k}) &= \sum_{I:0\le|I|\le\iota} g_{j,|I|}(m - |\boldsymbol{k}|_{I^*}) \\
&\le \sum_{i=0}^{\iota}\sum_{\ell=0}^{\iota} \eta(\ell, i, \boldsymbol{k}) g_{j,i}(2(\iota-\ell)+1) \\
&= \sum_{i=0}^{\iota}\sum_{\ell=0}^{\iota} p(\ell, i, \boldsymbol{k})\binom{j}{i} g_{j,i}(2(\iota-\ell)+1) \\
&= \sum_{i=0}^{\iota}\sum_{\ell=0}^{\iota-i} p(\ell, i, \boldsymbol{k})\binom{j}{i} g_{j,i}(2(\iota-\ell)+1) \qquad (40) \\
&\le \sum_{i=0}^{\iota}\binom{j}{i} g_{j,i}(2\iota+1) = G(\tilde{m}, \mathbf{0}),
\end{aligned}
$$

where $\tilde{m} = 2\iota + 1$. In the above, the first inequality is obtained by replacing $g_{j,|I|}(m - |\boldsymbol{k}|_{I^*})$ by $g_{j,i}(2(\iota-\ell)+1)$ for each of the $\eta(\ell, i, \boldsymbol{k})$ pairs $(m, \boldsymbol{k})$ dominated by $(2(\iota-\ell)+1, \mathbf{0})$ at I, where $|I| = i$; the third equality holds because if $\ell > \iota - i$, then $2(\iota-\ell)+1 < 2(\iota-(\iota-i))+1 = 2i+1$ and therefore $g_{j,i}(2(\iota-\ell)+1) = 0$. Similarly, $\ell \le \iota - i$ implies $2(\iota-\ell)+1 \ge 2i+1$ and so $g_{j,i}(2(\iota-\ell)+1) > 0$ in this case. The last inequality comes from applying Lemma 15, whose conditions hold because:

1. $\binom{j}{i} g_{j,i}(2(\iota-\ell)+1)$ corresponds to $x_{\ell+1,i+1}$ in Lemma 15;
2. Lemma 14 together with (37) shows the decreasing row-sums condition is satisfied, i.e., $G(2(\iota-\ell)+1, \mathbf{0}) = \sum_i x_{\ell+1,i+1}$ is decreasing with ℓ;
3. The increasing-within-column assumption of Lemma 15 is satisfied because the sum (40) only includes positive values of $g_{j,i}(2(\iota-\ell)+1)$ (as shown above), which in turn allows us to invoke Part 2 of Lemma 13;
4. $p(\ell, i, \boldsymbol{k})$ corresponds to $\alpha_{\ell+1,i+1}$ in Lemma 15;
5. To see that the $p(\ell, i, \boldsymbol{k})$'s obey the decreasing-cumulative-sums condition (51) in Lemma 15, we argue that our probabilistic interpretation of the $p(\ell, i, \boldsymbol{k})$ based on the sets defined in (39) should make it clear that for $i = 0, \ldots, \iota - 1$ and $0 \le r \le \iota$,
$$\sum_{\ell=0}^{r} p(\ell, i, \boldsymbol{k}) \ge \sum_{\ell=0}^{r} p(\ell, i+1, \boldsymbol{k}).$$

Therefore $G(m, \boldsymbol{k}) \le G(\tilde{m}, \mathbf{0})$, as required. □

Technical Lemmas

The following result [7, Lemma 2] is used to prove intermediate inequalities needed in our analysis.

Lemma 10 *Let $P_{m,n}(z)$ be the polynomial defined by*

$$P_{m,n}(z) = \sum_{j=0}^{m} \binom{n}{j} z^j, \qquad 0 < m < n-1,$$

Then for $z \neq -1$

$$\frac{P_{m,n}(z)}{(1+z)^n \binom{n}{m}(n-m)} = \int_{z/(z+1)}^{1} u^m (1-u)^{n-m-1} du. \tag{41}$$

We also need the following identity for integers $c > a \geq 0$, which may be found in [5, (5.16)]

$$\sum_{j=0}^{a} \binom{c}{j} (-1)^j = \binom{c-1}{a} (-1)^a. \tag{42}$$

We now state and prove a number of technical lemmas that were used within the above proofs. The first two lemmas are used to prove Lemma 5, and the next three are used for Lemmas 8 and 9.

Lemma 11 *For $b \geq s \geq 2$ and $0 \leq k < s$, let*

$$Q(b,k,s) := \sum_{j=0}^{k} (-1)^j \binom{s}{j} (b^{k-j} - 1).$$

Then

$$Q(b,k,s) \leq \begin{cases} b^k \left(\frac{b-1}{b}\right)^s & \text{if } k \text{ is even} \\ b^k \left(\frac{b-1}{b}\right)^s + \binom{s-1}{k} & \text{if } k > 1 \text{ is odd,} \\ (b-1) & \text{if } k = 1. \end{cases} \tag{43}$$

Proof The statement holds trivially for $k = 0$. For $k > 0$ we apply Lemma 10 and (42) to obtain

$$\begin{aligned} Q(b,k,s) &= b^k P_{k,s}(-1/b) - \binom{s-1}{k}(-1)^k \\ &= b^k \left(\frac{b-1}{b}\right)^s \int_{-1/(b-1)}^{1} u^k (1-u)^{s-k-1} c_{k,s} du - \binom{s-1}{k}(-1)^k \\ &= b^k \left(\frac{b-1}{b}\right)^s \left(\int_{-1/(b-1)}^{0} u^k (1-u)^{s-k-1} c_{k,s} du + 1 \right) - \binom{s-1}{k}(-1)^k \end{aligned}$$

where $c_{k,s}$ is the constant that makes the integrand a beta pdf with parameters $(k+1, s-k)$, i.e., $c_{k,s} = s\binom{s-1}{k}$.

Now, if k is odd then $\int_{-1/(b-1)}^{0} u^k(1-u)^{s-k-1}c_{k,s}du \le 0$ and thus we get

$$Q(b,k,s) \le b^k\left(\frac{b-1}{b}\right)^s + \binom{s-1}{k}.$$

Note also that when $k=1$, $Q(b,k,s) = b-1$ (which is not necessarily bounded from above by $b^k\left(\frac{b-1}{b}\right)^s + \binom{s-1}{k} = b((b-1)/b)^s + s - 1$. It is for this reason we treat the case $k=1$ separately). If k is even, then

$$\int_{\frac{-1}{b-1}}^{0} u^k(1-u)^{s-k-1}c_{k,s}du \le c_{k,s}\frac{1}{b-1}\frac{1}{(b-1)^k}\left(1+\frac{1}{b-1}\right)^{s-k-1} = c_{k,s}\frac{b^{s-k-1}}{(b-1)^s}.$$

Therefore when k is even

$$\begin{aligned} Q(b,k,s) &= b^k\left(\frac{b-1}{b}\right)^s\left(1 + c_{k,s}\frac{b^{s-k-1}}{(b-1)^s}\right) - \binom{s-1}{k} \\ &= b^k\left(\frac{b-1}{b}\right)^s + \left(\frac{s}{b}-1\right)\binom{s-1}{k} \le b^k\left(\frac{b-1}{b}\right)^s \end{aligned}$$

since $s \le b$. □

Lemma 12 *Consider a $(0, m, s)$-net in base b. Let $\emptyset \neq J \subset \{1, \ldots, s\}$, and $I \subseteq J$, with $I^* = I \cup J^c$. Then the following bounds hold:*
(i) if $m \ge |\boldsymbol{k}|_{I^*} + \boldsymbol{d}_J + c|I| + |I^{*c}|$, then

$$m_b(\boldsymbol{k}, \boldsymbol{d}, c, J, I; P_n) = b^{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J - c|I|-|I^{*c}|}(b-1)^{|I^{*c}|};$$

(ii) if $|\boldsymbol{k}|_{I^*} + |\boldsymbol{d}|_J + c|I| + 1 < m < |\boldsymbol{k}|_{I^*} + \boldsymbol{d}_J + c|I| + |I^{*c}|$ then

$$\begin{aligned} m_b(\boldsymbol{k}, \boldsymbol{d}, c, J, I; P_n) \le b^{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J}&\left(\frac{b-1}{b}\right)^{|I^{*c}|} \\ &+\binom{|I^{*c}|-1}{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J - c|I|}\mathbf{1}_{m-|\boldsymbol{k}|_{I^*}-|\boldsymbol{d}|_J-c|I|} \end{aligned}$$

where

$$\mathbf{1}_x = \begin{cases} 1 & \text{if } x \text{ is odd} \\ 0 & \text{otherwise.} \end{cases}$$

(iii) if $m = |\boldsymbol{k}|_{I^*} + |\boldsymbol{d}|_J + c|I| + 1$ then

$$m_b(\boldsymbol{k}, \boldsymbol{d}, c, J, I; P_n) = (b-1).$$

(iv) if $m \le |\boldsymbol{k}|_{I^*} + |\boldsymbol{d}|_J + c|I|$ *then* $m_b(\boldsymbol{k}, \boldsymbol{d}, c, J, I; P_n) = 0$.

Proof Using the quantities $n_b(\boldsymbol{k})$ defined in [10], their relation to $m_b(\boldsymbol{k}; P_n)$ and the value of the latter for a $(0, m, s)$-net, we write

$$\begin{aligned} m_b(\boldsymbol{k}, \boldsymbol{d}, c, J, I; P_n) &= \sum_{i_j \ge k_j, j \in J^c; i_j \ge k_j + d_j + c, j \in I} n_b(\boldsymbol{i}_{I^*} : \boldsymbol{d}_{I^{*c}}) \\ &= \sum_{\mathbf{e} \in \{0,1\}^{|I^{*c}|}} (-1)^{|\mathbf{e}|} m_b((\boldsymbol{k}_{J^c} : (\boldsymbol{k} + \boldsymbol{d} + 2)_I : (\mathbf{d} + \mathbf{e})_{I^{*c}}); P_n) \\ &= \sum_{j=0}^{|I^{*c}|} (-1)^j \binom{|I^{*c}|}{j} \max(b^{m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - c|I| - j} - 1, 0) \end{aligned} \tag{44}$$

where $(\boldsymbol{i}_I : \boldsymbol{d}_{I^c})$ represents the vector with jth component given by i_j if $j \in I$ and by d_j if $j \notin I$. If $m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - c|I| - |I^{*c}| \ge 0$ then the above sum is given by

$$\begin{aligned} m_b(\boldsymbol{k}, \boldsymbol{d}, c, J, I; P_n) &= b^{m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - c|I| - |I^{*c}|} \sum_{j=0}^{|I^{*c}|} (-1)^j \binom{|I^{*c}|}{j} b^{|I^{*c}| - j} \\ &= b^{m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - c|I| - |I^{*c}|} (b-1)^{|I^{*c}|}. \end{aligned}$$

If $m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - c|I| \le 0$ then the max inside the sum (44) always yields 0. When $1 < m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - c|I| < |I^{*c}|$, then (44) is given by

$$\begin{aligned} &\sum_{j=0}^{m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - c|I|} (-1)^j \binom{|I^{*c}|}{j} (b^{m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - c|I| - j} - 1) \\ &\le \left(b^{m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J} \left(\frac{b-1}{b} \right)^{|I^{*c}|} + \binom{|I^{*c}| - 1}{m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - c|I|} \mathbf{1}_{m - |\boldsymbol{k}|_I - |\boldsymbol{d}|_{I^c} - c|I|} \right), \end{aligned}$$

where the last inequality is obtained by applying Lemma 11 with $s = |I^c|$ and $k = m - |\boldsymbol{k}|_{I^*} - |\boldsymbol{d}|_J - c|I|$. Finally, when $m = |\boldsymbol{k}|_{I^*} + |\boldsymbol{d}|_J + c|I| + 1$, then (44) is given by

$$\sum_{j=0}^{1} (-1)^j \binom{|I^c|}{j} (b^{1-j} - 1) = b - 1.$$

□

Lemma 13 *The function* $g_{j,i}(\ell)$ *defined in* (16) *with* $0 \le i < j$ *and* $j \ge 1$ *satisfies the following properties.*

1. *For a given i, if $\ell \geq 1$ is odd then $g_{j,i}(\ell) \geq g_{j,i}(\ell+1)$.*
2. *For a given i, if $\ell > 2i$ is odd then $g_{j,i}(\ell) \geq g_{j,i}(\ell+r)$ for all $r \geq 0$.*

Proof For (1): if $\ell \geq j+i$ then $g_{j,i}(\ell) = g_{j,i}(\ell+1) = 1$; if $2i < \ell < j+i$, then $g_{j,i}(\ell) > 1$ while $g_{j,i}(\ell+1) = 1$; if $\ell \leq 2i$ then $g_{j,i}(\ell) = 0$ and since ℓ is odd, it means $\ell \leq 2i-1$, and thus $\ell+1 \leq 2i$, implying that $g_{j,i}(\ell+1) = 0$. For (2): if $\ell \geq j+i$ then $g_{j,i}(\ell+r) = 1$ for all $r \geq 0$; if $2i+1 < \ell < j+i$, then the function $g_{j,i}(\ell)$ is increasing as ℓ decreases over odd values strictly between $j+i$ and $2i+1$; this is because when ℓ decreases by 2, $h_{j,i}(\ell)$ increases by a factor of at least $2b^2/((j-i-1)(j-i-2))$, which is at least 1 since $j \leq s \leq b$. Finally, we need to show that $g_{j,i}(2i+1) = (b/(b-1))^{j-i-1} \geq g_{j,i}(2i+3)$, i.e., that

$$\left(\frac{b}{b-1}\right)^{j-i-1} \geq 1 + \frac{b^{j+i-(2i+3)}}{(b-1)^{j-i}}\binom{j-i-1}{2i+3-2i} = 1 + \left(\frac{b}{b-1}\right)^{j-i}\frac{1}{b^3}\binom{j-i-1}{3}.$$

Using the bound $((b-1)/b)^j \leq (b-j-1)/(b-1)$ shown in the proof of Lemma 14, we have that the above holds if

$$\begin{aligned} \frac{b-1}{b} &\geq \frac{(j-i-1)(j-i-2)(j-i-3)}{6b^3} + \frac{b-(j-i)-1}{b-1} \\ \Leftrightarrow \frac{(b-1)^2 - b(b-j+i-1)}{b(b-1)} &\geq \frac{(j-i-1)(j-i-2)(j-i-3)}{6b^3} \\ \Leftrightarrow 6b^2(1+b(j-i-1)) &\geq (b-1)(j-i-1)(j-i-2)(j-i-3), \end{aligned}$$

which is clearly true since $j \leq s \leq b$ and $i \geq 0$ and therefore $6b^3(j-i-1) \geq (b-1)(j-i-1)(j-i-2)(j-i-3)$. □

Lemma 14 *Let $s \geq 3$ and $b \geq s$. Let m be odd with $1 \leq m \leq 2s-3$, and consider the function*

$$G(m,s) = \sum_{j=0}^{0.5(m-3)} \binom{s}{j} + \sum_{j=\max(0,m-s+1)}^{0.5(m-3)} \binom{s}{j} h_{s,j}(m) + \binom{s}{0.5(m-1)}\left(\frac{b}{b-1}\right)^{s-0.5(m-1)-1},$$

where $h_{s,j}(m)$ is as defined in (15), i.e.,

$$h_{s,j}(m) = \binom{s-j-1}{m-2j}\frac{b^{s+j-m}}{(b-1)^{s-m}}.$$

Then $G(m,s) \geq G(m-2,s)$ for $m \geq 3$ odd. That is, $G(m,s)$ is decreasing over the odd integers from $2s-3$ down to 3.

Proof First, we compute

$$\begin{aligned}
&\left(\frac{b}{b-1}\right)^{s-0.5(m-1)-1} - h_{s,0.5(m-1)}(m) \\
&= \left(\frac{b}{b-1}\right)^{s-0.5(m-1)-1} - \binom{s-0.5(m-1)-1}{1} \frac{b^{0.5(m-1)+s-m}}{(b-1)^{s-0.5(m-1)}} \\
&= \left(\frac{b}{b-1}\right)^{s-0.5(m-1)} \frac{b-1-(s-0.5(m-1)-1)}{b} \\
&= \left(\frac{b}{b-1}\right)^{s-0.5(m-1)} \frac{b-(s-0.5(m-1))}{b}.
\end{aligned}$$

Using this, we can write

$$\begin{aligned}
G(m,s) = &\sum_{j=0}^{0.5(m-3)} \binom{s}{j} + \sum_{j=\max(0,m-s+1)}^{0.5(m-1)} \binom{s}{j} h_{s,j}(m) \\
&+ \binom{s}{0.5(m-1)} \left(\frac{b}{b-1}\right)^{s-0.5(m-1)} \frac{0.5(m-1)+b-s}{b}. \qquad (45)
\end{aligned}$$

Next, we show that for $2 \le j \le 0.5(m-1)$:

$$\binom{s}{j} h_{s,j}(m) \ge \binom{s}{j-2} h_{s,j-2}(m-2). \qquad (46)$$

$$\begin{aligned}
&\binom{s}{j} h_{s,j}(m) - \binom{s}{j-2} h_{s,j-2}(m-2) \\
&= \binom{s}{j}\binom{s-j-1}{m-2j} \frac{b^{s+j-m}}{(b-1)^{s-j}} - \binom{s}{j-2}\binom{s-(j-2)-1}{m-2-2(j-2)} \frac{b^{s+j-2-(m-2)}}{(b-1)^{s-(j-2)}} \\
&= \binom{s}{j}\binom{s-j-1}{m-2j} \frac{b^{s+j-m}}{(b-1)^{s-j}} - \binom{s}{j-2}\binom{s-j+1}{m-2j+2} \frac{b^{s+j-m}}{(b-1)^{s-j+2}} \\
&= \binom{s}{j}\binom{s-j-1}{m-2j} \frac{b^{s+j-m}}{(b-1)^{s-j}} \\
&\quad \cdot \left(1 - \frac{j(j-1)}{(s-j+2)(s-j+1)} \frac{(s-j+1)(s-j)}{(m-2j+2)(m-2j+1)} \frac{1}{(b-1)^2}\right).
\end{aligned}$$

Hence to prove (46), we need to show that

$$1 \ge \frac{j(j-1)}{(s-j+2)(s-j+1)} \frac{(s-j+1)(s-j)}{(m-2j+2)(m-2j+1)} \frac{1}{(b-1)^2},$$

which holds because

$$\frac{j(j-1)}{(s-j+2)(s-j+1)}\frac{(s-j+1)(s-j)}{(m-2j+2)(m-2j+1)}\frac{1}{(b-1)^2} \le \frac{j(j-1)}{6}\frac{1}{(b-1)^2}$$
$$\le \frac{(s-2)(s-3)}{6(b-1)^2} \le \frac{1}{6} \le 1,$$

since $j \le 0.5(m-1) \le s-2$ and $b \ge s$.

Using (45), $G(m,s) \ge G(m-2,s)$ can be shown to hold if

$$\begin{aligned}
&\sum_{j=0}^{0.5(m-3)} \binom{s}{j} + \sum_{j=\max(0,m-s+1)}^{0.5(m-1)} \binom{s}{j} h_{s,j}(m) \\
&\quad + \binom{s}{0.5(m-1)} \left(\frac{b}{b-1}\right)^{s-0.5(m-1)} \frac{0.5(m-1)+b-s}{b} \\
&\ge \sum_{j=0}^{0.5(m-5)} \binom{s}{j} + \sum_{j=\max(0,m-2-s+1)}^{0.5(m-3)} \binom{s}{j} h_{s,j}(m-2) \\
&\quad + \binom{s}{0.5(m-3)} \left(\frac{b}{b-1}\right)^{s-0.5(m-3)} \frac{0.5(m-3)+b-s}{b}. \qquad (47)
\end{aligned}$$

In turn, using (46), we know that:

$$\sum_{j=\max(0,m-s+1)}^{0.5(m-1)} \binom{s}{j} h_{s,j}(m) \ge \sum_{j=\max(0,m-2-s+1)}^{0.5(m-5)} \binom{s}{j} h_{s,j}(m-2)$$

and therefore to show (47) it is sufficient to show that

$$\begin{aligned}
&\binom{s}{0.5(m-3)} + \binom{s}{0.5(m-1)} \left(\frac{b}{b-1}\right)^{s-0.5(m-1)} \frac{0.5(m-1)+b-s}{b} \\
&\ge \binom{s}{0.5(m-3)} h_{s,0.5(m-3)}(m-2) \\
&\quad + \binom{s}{0.5(m-3)} \left(\frac{b}{b-1}\right)^{s-0.5(m-3)} \frac{0.5(m-3)+b-s}{b}, \qquad (48)
\end{aligned}$$

where

$$\begin{aligned}
h_{s,0.5(m-3)}(m-2) &= \binom{s-0.5(m-3)-1}{m-2-(m-3)} \frac{b^{s+0.5(m-3)-(m-2)}}{(b-1)^{s-0.5(m-3)}} \\
&= (s-0.5(m-1)) \left(\frac{b}{b-1}\right)^{s-0.5(m-1)} \frac{1}{b-1}.
\end{aligned}$$

The following inequality will be helpful in this proof:

Claim 1 For $b \geq 2$ and $j \geq 1$, we have that

$$\left(\frac{b}{b-1}\right)^{j} \leq \frac{b-1}{b-(j+1)}. \tag{49}$$

Proof The inequality is equivalent to having

$$(b-1)^{j+1} \geq b^{j}(b-(j+1)).$$

Applying the mean value theorem to $f(x) = x^{j+1}$ and noticing $f'(x)$ is monotone increasing for $x \geq 0$, we get that $f(b) - f(b-1) = f'(\xi) \leq f'(b)$ for some $\xi \in (b-1, b)$ and thus $(b-1)^{j+1} \geq b^{j+1} - (j+1)b^{j}$. □

Going back to our goal of proving (48), it is sufficient to show that

$$\begin{aligned}
&1 + \left(\frac{b}{b-1}\right)^{s-0.5(m-1)} \frac{s-0.5(m-3)}{0.5(m-1)} \frac{0.5(m-1)+b-s}{b} \\
&\geq \left(\frac{b}{b-1}\right)^{s-0.5(m-1)} \left(\frac{s-0.5(m-1)}{b-1} + \frac{b}{b-1}\frac{0.5(m-3)+b-s)}{b}\right) \\
\Leftrightarrow &\left(\frac{b-1}{b}\right)^{s-0.5(m-1)} + \frac{s-0.5(m-3)}{0.5(m-1)} \frac{0.5(m-1)+b-s}{b} \\
&\geq \left(\frac{s-0.5(m-1)}{b-1} + \frac{0.5(m-3)+b-s}{b-1}\right).
\end{aligned} \tag{50}$$

Using (49) to simplify the LHS of (50), we see that (50) holds if

$$\begin{aligned}
&\frac{b-(s-0.5(m-1))-1}{b-1} + \frac{s-0.5(m-3)}{0.5(m-1)} \frac{0.5(m-1)+b-s}{b} \\
&\geq \left(\frac{s-0.5(m-1)}{b-1} + \frac{0.5(m-3)+b-s}{b-1}\right) \\
\Leftrightarrow &\frac{s-0.5(m-3)}{0.5(m-1)} \frac{0.5(m-1)+b-s}{b} \geq \frac{s-0.5(m-1)}{b-1} \\
\Leftrightarrow &\frac{b-1}{b} \geq \frac{s-0.5(m-1)}{s-0.5(m-3)} \frac{0.5(m-1)}{0.5(m-1)+b-s},
\end{aligned}$$

which holds because $s - 0.5(m-3) \leq s \leq b$ and $\frac{b-1}{b} \geq \frac{b-1-x}{b-x}$ if $x \geq 0$.

Definition 3 We denote by $\mathcal{A}_w$ the set of $w \times \ell$ *weight matrices* A with entries $\alpha_{i,j} \geq 0$ that satisfy the following two conditions: 1) $\sum_{i=1}^{w} \alpha_{i,j} = 1$ for all $j = 1, \ldots, \ell$; 2) the weights $\alpha_{i,j}$ obey a *decreasing-cumulative-sums* condition as follows: for $1 \leq i \leq w$, $1 \leq j \leq \ell$, let

$$A_{i,j} = \sum_{k=1}^{i} \alpha_{k,j}. \tag{51}$$

Then $A_{i,j} \geq A_{i,j+1}$ for each $i = 1, \ldots, w$ and $j = 1, \ldots, \ell - 1$ (when $i = w$ we have $A_{w,j} = 1$ for all j). This means the weights on the first row are decreasing from left to right; the partial sums of the two first rows are decreasing from left to right, etc.

Lemma 15 *Let X be a $w \times \ell$ matrix with $\ell \geq w$ and entries $x_{i,j} \geq 0$ and of the form*

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,\ell-w+1} & \cdots & & x_{1,\ell-1} & x_{1,\ell} \\ x_{2,1} & \cdots & x_{2,\ell-w+1} & \cdots & & x_{2,\ell-1} & 0 \\ \vdots & & \vdots & & \iddots & \iddots & \\ x_{w-1,1} & \cdots & x_{w-1,\ell-w+1} & x_{w-1,\ell-w+2} & 0 & \cdots & \\ x_{w,1} & \cdots & x_{w,\ell-w+1} & 0 & \cdots & & 0 \end{bmatrix}.$$

that is $x_{i,j} > 0$ if and only if $i + j \leq \ell + 1$, for $1 \leq i \leq w$, $1 \leq j \leq \ell$. We assume X satisfies the following two conditions: first,

$$\sum_{j=1}^{\ell} x_{1,j} \geq \sum_{j=1}^{\ell-1} x_{2,j} \geq \cdots \geq \sum_{j=1}^{\ell-w+1} x_{w,j}$$

(we refer to this as the decreasing-row-sums *condition) and second,*

$$x_{1,j} \leq x_{2,j} \leq \cdots \leq x_{\min(w,\ell-i+1),j}, \qquad j = 1, \ldots, \ell \tag{52}$$

(we refer to this as the increasing-within-column *condition).*

Let A be a weight matrix in $\mathcal{A}_w$ and let

$$\|A \circ X\|_1 = \sum_{j=1}^{\ell} \alpha_{1,j} x_{1,j} + \sum_{j=1}^{\ell-1} \alpha_{2,j} x_{2,j} + \cdots + \sum_{j=1}^{\ell-w+1} \alpha_{w,j} x_{w,j}.$$

Then for any $A \in \mathcal{A}_w$

$$\|A \circ X\|_1 \leq \sum_{j=1}^{\ell} x_{1,j}. \tag{53}$$

That is, the weight matrix $A \in \mathcal{A}_w$ that maximize the LHS of (53) is the one with 1's on the first row and 0's elsewhere.

Proof First, note that $A \in \mathcal{A}_w$ implies that cumulative sums from the last row up are increasing, i.e., for $R_{i,j} = \sum_{k=i}^{w} \alpha_{k,j}$, we have $R_{i,j} \leq R_{i,j+1}$ for $j = 1, \ldots, \ell - 1$.

We proceed by induction on $w \geq 2$.

If $w = 2$, then it suffices to show that for $A \in \mathcal{A}_2$, we have that

$$\sum_{j=1}^{\ell} \alpha_{1,j} x_{1,j} + \sum_{j=1}^{\ell-1} (1-\alpha_{1,j}) x_{2,j} \le \sum_{i=1}^{\ell} x_{1,j} \Leftrightarrow \sum_{j=1}^{\ell-1} (1-\alpha_{1,j}) x_{2,j} \le \sum_{j=1}^{\ell} (1-\alpha_{1,j}) x_{1,j},$$

or, equivalently, that

$$\sum_{j=1}^{\ell-1} (1-\alpha_{1,j})(x_{2,j} - x_{1,j}) \le (1-\alpha_{1,\ell}) x_{1,\ell}.$$

Now, we know that

$$\sum_{j=1}^{\ell} x_{1,j} \ge \sum_{j=1}^{\ell-1} x_{2,j} \Leftrightarrow \sum_{j=1}^{\ell-1} (x_{2,j} - x_{1,j}) \le x_{1,\ell}$$

with $x_{2,j} - x_{1,j} \ge 0$. Therefore

$$\sum_{j=1}^{\ell-1} (1-\alpha_{1,j})(x_{2,j} - x_{1,j}) \le (1-\alpha_{1,\ell}) \sum_{j=1}^{\ell-1} (x_{2,j} - x_{1,j}) \le (1-\alpha_{1,\ell}) x_{1,\ell},$$

where the first inequality holds because the $\alpha_{1,j}$'s are decreasing.

Now assume the statement holds for $w - 1 \ge 2$. First we create a new weight matrix $\tilde{A}$ by merging the two last rows into the second-to-last one and setting the last one to zero, i.e., we define $\tilde{\alpha}_{w-1,j}$ as

$$\begin{aligned} \tilde{\alpha}_{w-1,j} &= \alpha_{w-1,j} + \alpha_{w,j} \qquad j = 1, \ldots, \ell \\ \tilde{\alpha}_{w,j} &= 0 \qquad j = 1, \ldots, \ell \\ \tilde{\alpha}_{i,j} &= \alpha_{i,j}, i = 1, \ldots, w-2, j = 1, \ldots, \ell. \end{aligned}$$

With this change, we claim that $\tilde{A} \in \mathcal{A}_w$. Indeed:

1. $\tilde{\alpha}_{i,j} \ge 0$
2. $\sum_{i=1}^{w} \tilde{\alpha}_{i,j} = \sum_{i=1}^{w-2} \alpha_{i,j} + (\alpha_{w-1,j} + \alpha_{w,j}) + 0 = 1.$
3. $\tilde{A}_{i,j} = A_{i,j}$ for $i = 1, \ldots, w-2$ and $\tilde{A}_{w-1,j} = A_{w,j} = 1$ for $j = 1, \ldots, \ell$.

Next, we show that

$$\|\tilde{A} \circ X\|_1 \ge \|A \circ X\|_1. \tag{54}$$

Since $\alpha_{i,j} = \tilde{\alpha}_{i,j}$ for $i < w - 1$, then (54) holds if and only if

$$\sum_{j=1}^{\ell-w+2}(\alpha_{w-1,j}+\alpha_{w,j})x_{w-1,j} \geq \sum_{j=1}^{\ell-w+2}\alpha_{w-1,j}x_{w-1,j} + \sum_{j=1}^{\ell-w+1}\alpha_{w,j}x_{w,j}$$

$$\Leftrightarrow \sum_{j=1}^{\ell-w+1}\alpha_{w,j}x_{w-1,j} + \alpha_{w,\ell-w+2}x_{w-1,\ell-w+2} \geq \sum_{j=1}^{\ell-w+1}\alpha_{w,j}x_{w,j}$$

$$\Leftrightarrow \sum_{j=1}^{\ell-w+1}\alpha_{w,j}(x_{w,j}-x_{w-1,j}) \leq \alpha_{w,\ell-w+2}x_{w-1,\ell-w+2}.$$

By the decreasing-row-sum assumption on the $x_{i,j}$'s we know that

$$0 \leq \sum_{j=1}^{\ell-w+1}(x_{w,j}-x_{w-1,j}) \leq x_{w-1,\ell-w+2}$$

and by assumption that $A \in \mathcal{A}_w$ we have that $\alpha_{w,1} \leq \alpha_{w,2} \leq \cdots \leq \alpha_{w,\ell}$. Therefore

$$\sum_{j=1}^{\ell-w+1}\alpha_{w,j}(x_{w,j}-x_{w-1,j}) \leq \alpha_{w,\ell-w+1}\sum_{j=1}^{\ell-w+1}(x_{w,j}-x_{w-1,j})$$
$$\leq \alpha_{w,\ell-w+1}x_{w-1,\ell-w+2} \leq \alpha_{w,\ell-w+2}x_{w-1,\ell-w+2},$$

as required to show that (54) holds.

Next, to use the induction hypothesis, we observe that $\tilde{\alpha}_{w,j} = 0$ implies we can essentially ignore the $x_{w,j}$'s. More formally, let $\tilde{A}_{w-1}$ be the matrix formed by the first $w-1$ rows of $\tilde{A}$ and similarly for X_{w-1}. Then $\tilde{A}_{w-1} \in \mathcal{A}_{w-1}$, since

1. $\tilde{\alpha}_{i,j} \geq 0$ for $i = 1, \ldots, w-1$, $j = 1, \ldots, \ell$
2. $\sum_{i=1}^{w-1}\tilde{\alpha}_{i,j} = \sum_{i=1}^{w}\alpha_{i,j} = 1$ for $j = 1, \ldots, \ell$
3. $\tilde{A}_{i,j} \geq \tilde{A}_{i,j+1}$ as verified earlier (and note that $\tilde{\alpha}_{w-1,1} \leq \cdots \leq \tilde{\alpha}_{w-1,\ell}$ by assumption that $A \in \mathcal{A}_w$ and since $\tilde{\alpha}_{w-1,j} = R_{w-1,j}$.)

By applying the induction hypothesis, we obtain

$$\|\tilde{A}_{w-1} \circ X_{w-1}\|_1 \leq \sum_{j=1}^{\ell}x_{1,j}$$

and since $\|A \circ X\|_1 \leq \|\tilde{A} \circ X\|_1 = \|\tilde{A}_{w-1} \circ X_{w-1}\|$, this proves the result. □

References

1. Dick, J., Pillichshammer, F.: Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration. Cambridge University Press, UK (2010)
2. Doerr, B., Gnewuch, M.: On Negative Dependence Properties of Latin Hypercube Samples and Scrambled Nets (2021). Preprint on arXiv.org
3. Gnewuch, M., Wnuk, M., Hebbinghaus, N.: On negatively dependent sampling schemes, variance reduction, and probabilistic upper discrepancy bounds. In: Bylik, D., Dick, J., Pillichshammer, F. (eds.) Discrepancy Theory, Radon Series on Computational and Applied Mathematics, vol. 26, pp. 43–68, De Gruyter (2020)
4. Gnewuch, M., Hebbinghaus, N.: Discrepancy Bounds for a Class of Negatively Dependent Random Points Including Latin Hypercube Samples (2021). Preprint on arXiv.org
5. Graham, R., Knuth, D., Patashnik, O.: Concrete Mathematics: A Foundation for Computer Science, Addison-Wesley (1989)
6. Niederreiter, H.: Random number generation and Quasi-Monte Carlo methods. In: SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 63. SIAM, Philadelphia (1992)
7. Ostrovskii, I.V.: On a problem of A. Eremenko. Comput. Meth. Funct. Th. **4**, 275–282 (2004)
8. Owen, A.B.: Randomly permuted (t, m, s)-nets and (t, s)-sequences. In: Niederreiter, H., Shiue, P.J.-S. (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing. Lecture Notes in Statistics, vol. 106, pp. 299–317. Springer, New York (1995)
9. Owen, A.B.: Scrambling Sobol' and Niederreiter-Xing points. J. Complex. **14**, 466–489 (1998)
10. Wiart, J., Lemieux, C., Dong, G.: On the dependence structure and quality of scrambled (t, m, s)-nets. Monte Carlo Methods Appl. **27**, 1–26 (2021)
11. Wnuk, M., Gnewuch, M.: Note on pairwise negative dependence of randomized rank-1 lattices. Oper. Res. Lett. **48**, 410–414 (2020)

Lower Bounds for the Number of Random Bits in Monte Carlo Algorithms

Stefan Heinrich

Abstract We continue the study of restricted Monte Carlo algorithms in a general setting. Here we show a lower bound for minimal errors in the setting with finite restriction in terms of deterministic minimal errors. This generalizes a result of Heinrich, Novak, and Pfeiffer (in: Monte Carlo and Quasi-Monte Carlo Methods 2002, H. Niederreiter, ed., Springer-Verlag, Berlin, 2004, pp. 27–49) to the adaptive setting. As a consequence, the lower bounds on the number of random bits from that paper also hold in this setting. We also derive a lower bound on the number of needed bits for integration of Lipschitz functions over the Wiener space, complementing a result of Giles, Hefter, Mayer, and Ritter (J. Complexity 54 (2019), 101395).

Keywords Random bit · Complexity theory · Lower bounds · Integration · Wiener space

1 Introduction

Restricted Monte Carlo algorithms were considered in [3–6, 11–14, 16, 17]. Restriction usually means that the algorithm has access only to random bits or to random variables with finite range. Most of these papers on restricted randomized algorithms consider the non-adaptive case. Only [5] includes adaptivity, but considers a class of algorithms where each information call is followed by one random bit call.

A general definition restricted Monte Carlo algorithms was given in [10]. It extends the previous notions in two ways: Firstly, it includes full adaptivity (in particular, an information call must not necessarily be followed by a random bit call), and secondly, it includes models in which the algorithms have access to an arbitrary, but fixed set of random variables, for example, uniform distributions on [0, 1]. In [10] the relation of restricted to unrestricted randomized algorithms was studied. In particular, it was shown that for each such restricted setting there is a computational problem that can be solved in the unrestricted randomized setting but not under the restriction.

S. Heinrich (✉)
Department of Computer Science, University of Kaiserslautern, 67653 Kaiserslautern, Germany
e-mail: heinrich@informatik.uni-kl.de

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_6

The aim of the present paper is to continue the study of the restricted setting. The main result is a lower bound for minimal errors in the setting with a finite restriction in terms of deterministic minimal errors. This generalizes a corresponding result from [11], see Proposition 1 there, to the adaptive setting with arbitrary finite restriction. The formal proof in this setting is technically more involved. As a consequence the lower bounds on the number of random bits from [11] also hold in this setting. Another corollary concerns integration of Lipschitz functions over the Wiener space [5]. It shows that the number of random bits used in the algorithm from [5] is optimal, up to logarithmic factors.

2 Restricted Randomized Algorithms in a General Setting

We work in the framework of information-based complexity theory (IBC) [13, 15], using specifically the general approach from [7, 8]. We recall the notion of a restricted randomized algorithm as recently introduced in [10]. This section is kept general, for specific examples illustrating this setup we refer to the integration problem considered in [10] as well as to the problems studied in Sect. 4.

We consider an abstract numerical problem

$$\mathcal{P} = (F, G, S, K, \Lambda), \tag{1}$$

where F and K are a non-empty sets, G is a Banach space, S a mapping from F to G, and Λ a nonempty set of mappings from F to K. The operator S is understood to be the solution operator that sends the input $f \in F$ to the exact solution $S(f)$ and Λ is the set of information functionals about the input $f \in F$ that can be exploited by an algorithm.

A probability space with access restriction is a tuple

$$\mathcal{R} = \big((\Omega, \Sigma, \mathbb{P}), K', \Lambda'\big), \tag{2}$$

with $(\Omega, \Sigma, \mathbb{P})$ a probability space, K' a non-empty set, and Λ' a non-empty set of mappings from Ω to K', the set of randomization functionals, supplying the admissible randomness to the algorithm (e.g., random bits, see (8) below). Define

$$\bar{K} = K \dot{\cup} K', \quad \bar{\Lambda} = \Lambda \dot{\cup} \Lambda',$$

where $\dot{\cup}$ is the disjoint union, and for $\lambda \in \bar{\Lambda}$, $f \in F$, $\omega \in \Omega$ we set

$$\lambda(f, \omega) = \begin{cases} \lambda(f) \text{ if } & \lambda \in \Lambda \\ \lambda(\omega) \text{ if } & \lambda \in \Lambda'. \end{cases}$$

An $\mathcal{R}$-restricted randomized algorithm for problem $\mathcal{P}$ is a tuple

$$A = ((L_i)_{i=1}^{\infty}, (\tau_i)_{i=0}^{\infty}, (\varphi_i)_{i=0}^{\infty})$$

such that $L_1 \in \bar{\Lambda}$, $\tau_0 \in \{0, 1\}$, $\varphi_0 \in G$, and for $i \in \mathbb{N}$

$$L_{i+1} : \bar{K}^i \to \bar{\Lambda}, \quad \tau_i : \bar{K}^i \to \{0, 1\}, \quad \varphi_i : \bar{K}^i \to G$$

are any mappings. Given $f \in F$ and $\omega \in \Omega$, we define $(\lambda_i)_{i=1}^{\infty}$ with $\lambda_i \in \bar{\Lambda}$ as follows:

$$\lambda_1 = L_1, \quad \lambda_i = L_i(\lambda_1(f, \omega), \ldots, \lambda_{i-1}(f, \omega)) \quad (i \geq 2). \tag{3}$$

If $\tau_0 = 1$, we define

$$\mathrm{card}_{\bar{\Lambda}}(A, f, \omega) = \mathrm{card}_{\Lambda}(A, f, \omega) = \mathrm{card}_{\Lambda'}(A, f, \omega) = 0.$$

If $\tau_0 = 0$, let $\mathrm{card}_{\bar{\Lambda}}(A, f, \omega)$ be the first integer $n \geq 1$ with

$$\tau_n(\lambda_1(f, \omega), \ldots, \lambda_n(f, \omega)) = 1,$$

if there is such an n. If $\tau_0 = 0$ and no such $n \in \mathbb{N}$ exists, put $\mathrm{card}_{\bar{\Lambda}}(A, f, \omega) = \infty$. Furthermore, set

$$\mathrm{card}_{\Lambda}(A, f, \omega) = |\{k \leq \mathrm{card}_{\bar{\Lambda}}(A, f, \omega) : \lambda_k \in \Lambda\}|$$
$$\mathrm{card}_{\Lambda'}(A, f, \omega) = |\{k \leq \mathrm{card}_{\bar{\Lambda}}(A, f, \omega) : \lambda_k \in \Lambda'\}|.$$

We have $\mathrm{card}_{\bar{\Lambda}}(A, f, \omega) = \mathrm{card}_{\Lambda}(A, f, \omega) + \mathrm{card}_{\Lambda'}(A, f, \omega)$. The output $A(f, \omega)$ of algorithm A at input (f, ω) is defined as

$$A(f, \omega) = \begin{cases} \varphi_0 & \text{if} \quad \mathrm{card}_{\bar{\Lambda}}(A, f, \omega) \in \{0, \infty\} \\ \varphi_n(\lambda_1(f, \omega), \ldots, \lambda_n(f, \omega)) & \text{if} \quad 1 \leq \mathrm{card}_{\bar{\Lambda}}(A, f, \omega) = n < \infty. \end{cases} \tag{4}$$

Informally speaking, the algorithm starts by evaluating an information functional $L_1 \in \Lambda$ at input $f \in F$, that is $L_1(f) \in K$, or by evaluating a randomization functional $L_1 \in \Lambda'$ at $\omega \in \Omega$, thus producing an instance of admissible randomness $L_1(\omega) \in K'$. Next a termination function $\tau_1(L_1(f, \omega))$ is evaluated. If its value is 1, the process is stopped. If the value is 0, we go on and choose, depending on $L_1(f, \omega)$, another functional $L_2 \in \Lambda \dot{\cup} \Lambda'$, then $L_2(f, \omega)$ is evaluated, and so on. The process is terminated as soon as $\tau_n(L_1(f, \omega), \ldots, L_n(f, \omega)) = 1$ for some n. Finally, the mapping $\varphi_n : K^n \to G$ is applied to the obtained string of functional values $(L_1(f, \omega), \ldots, L_n(f, \omega))$, representing the computations that lead to the approximation $A(f, \omega)$ to $S(f)$ in G.

Thus, in contrast to an unrestricted randomized algorithm (see the definition, e.g., in [7, 8, 10]), a restricted randomized algorithm can access the randomness of $(\Omega, \Sigma, \mathbb{P})$ only through the functionals $\lambda(\omega)$ for $\lambda \in \Lambda'$.

The set of all $\mathcal{R}$-restricted randomized algorithms for $\mathcal{P}$ is denoted by $\mathcal{A}^{\rm ran}(\mathcal{P}, \mathcal{R})$. Let $\mathcal{A}^{\rm ran}_{\rm meas}(\mathcal{P}, \mathcal{R})$ be the subset of those $A \in \mathcal{A}^{\rm ran}(\mathcal{P}, \mathcal{R})$ with the following properties: For each $f \in F$ the mappings

$$\omega \to \mathrm{card}_\Lambda(A, f, \omega) \in \mathbb{N}_0 \cup \{\infty\}, \quad \omega \to \mathrm{card}_{\Lambda'}(A, f, \omega) \in \mathbb{N}_0 \cup \{\infty\}$$

(and hence $\omega \to \mathrm{card}_{\tilde{\Lambda}}(A, f, \omega)$) are Σ-measurable and the mapping $\omega \to A(f, \omega) \in G$ is Σ-to-Borel measurable and $\mathbb{P}$-almost surely separably valued, the latter meaning that there is a separable subspace $G_f \subset G$ such that $\mathbb{P}(\{\omega \in \Omega : A(f, \omega) \in G_f\}) = 1$. The error of $A \in \mathcal{A}^{\rm ran}_{\rm meas}(\mathcal{P}, \mathcal{R})$ is defined as

$$e(\mathcal{P}, A) = \sup_{f \in F} \mathbb{E}\, \|S(f) - A(f, \omega)\|_G. \tag{5}$$

Given $n, k \in \mathbb{N}_0$, we define $\mathcal{A}^{\rm ran}_{n,k}(\mathcal{P}, \mathcal{R})$ to be the set of those $A \in \mathcal{A}^{\rm ran}_{\rm meas}(\mathcal{P}, \mathcal{R})$ satisfying for each $f \in F$

$$\mathbb{E}\, \mathrm{card}_\Lambda(A, f, \omega) \le n, \quad \mathbb{E}\, \mathrm{card}_{\Lambda'}(A, f, \omega) \le k.$$

The (n, k)-th minimal $\mathcal{R}$-restricted randomized error of S is defined as

$$e^{\rm ran}_{n,k}(\mathcal{P}, \mathcal{R}) = \inf_{A \in \mathcal{A}^{\rm ran}_{n,k}(\mathcal{P}, \mathcal{R})} e(\mathcal{P}, A). \tag{6}$$

Hence, $e^{\rm ran}_{n,k}(\mathcal{P}, \mathcal{R})$ is the minimal error among all $\mathcal{R}$-restricted randomized algorithms that use, on the average, not more than n information functionals and k randomization functionals.

Special cases of access restrictions are the following: An access restriction $\mathcal{R}$ is called finite, if

$$|K'| < \infty, \quad \lambda^{-1}(\{u\}) \in \Sigma \quad (\lambda' \in \Lambda', u \in K'). \tag{7}$$

In this case any $\mathcal{R}$-restricted randomized algorithm satisfies the following. For fixed $i \in \mathbb{N}_0$ and $f \in F$ the functions (see (3))

$$\omega \to L_i(\lambda_1(f, \omega), \ldots, \lambda_{i-1}(f, \omega)) \in \overline{\Lambda}, \quad \omega \to \lambda_i(f, \omega) \in \overline{K}$$

take finitely many values and are Σ-to-$\Sigma_0(\overline{\Lambda})$-measurable (respectively Σ-to-$\Sigma_0(\overline{K})$-measurable), where $\Sigma_0(M)$ denotes the σ-algebra generated by the finite subsets of a set M. This is readily checked by induction. It follows that the mapping

$$\omega \to \tau_i(\lambda_1(f, \omega), \ldots, \lambda_i(f, \omega)) \in \{0, 1\}$$

is measurable and

$$\omega \to \varphi_i(\lambda_1(f, \omega), \ldots, \lambda_i(f, \omega)) \in G$$

takes only finitely many values and is Σ-to-Borel-measurable. Consequently, for each $f \in F$ the functions $\mathrm{card}(A, f, \omega)$ and $\mathrm{card}'(A, f, \omega)$ are Σ-measurable, $A(f, \omega)$ takes only countably many values and is Σ-to-Borel-measurable, hence $A \in \mathcal{A}^{\mathrm{ran}}_{\mathrm{meas}}(\mathcal{P}, \mathcal{R})$.

An access restriction is called bit restriction, if

$$|K'| = 2, \quad \Lambda' = \{\xi_j : j \in \mathbb{N}\} \tag{8}$$

with $\xi_j : \Omega \to K' = \{u_0, u_1\}$ an independent sequence of random variables such that

$$P(\{\xi_j = u_0\}) = P(\{\xi_j = u_1\}) = 1/2, \quad (j \in \mathbb{N}). \tag{9}$$

The corresponding restricted randomized algorithms are called bit Monte Carlo algorithms. A non-adaptive version of these was considered in [3, 11, 14, 17].

Most frequently used is the case of uniform distributions on $[0, 1]$. This means $K' = [0, 1]$ and $\Lambda' = \{\eta_j : j \in \mathbb{N}\}$, with (η_j) being independent uniformly distributed on $[0, 1]$ random variables over $(\Omega, \Sigma, \mathbb{P})$.

We also use the notion of a deterministic and of an (unrestricted) randomized algorithm and the corresponding notions of minimal errors. For this we refer to [7, 8] as well as to Sect. 2 of [10]. Let us however mention that the definition of a deterministic algorithm follows a similar scheme as the one given above. More than that, we can give an equivalent definition of a deterministic algorithm, viewing it as a special case of a randomized algorithm with an arbitrary restriction $\mathcal{R}$. Namely, a deterministic algorithm is an $\mathcal{R}$-restricted randomized algorithm A with

$$L_1 \in \Lambda, \quad L_{i+1}(K^i) \subseteq \Lambda \quad (i \in \mathbb{N}).$$

Consequently, for each $f \in F$ and $\omega, \omega_1 \in \Omega$ we have $\mathrm{card}_{\Lambda'}(A, f, \omega) = 0$ and

$$\begin{aligned} A(f) &:= A(f, \omega) = A(f, \omega_1) \\ \mathrm{card}(A, f) &:= \mathrm{card}_{\tilde{\Lambda}}(A, f, \omega) = \mathrm{card}_{\Lambda}(A, f, \omega) = \mathrm{card}_{\Lambda}(A, f, \omega_1). \end{aligned}$$

Thus, such an algorithm ignores $\mathcal{R}$ completely. For a deterministic algorithm A relation (5) turns into

$$e(\mathcal{P}, A) = \sup_{f \in F} \|S(f) - A(f)\|_G. \tag{10}$$

A deterministic algorithm is in $\mathcal{A}^{\mathrm{ran}}_{n,k}(\mathcal{P}, \mathcal{R})$ iff $\sup_{f \in F} \mathrm{card}(A, f) \le n$. Taking the infimum in (6) over all such A gives the n-th minimal error in the deterministic setting $e^{\mathrm{det}}_n(\mathcal{P})$. Clearly, $e(\mathcal{P}, A)$ and $e^{\mathrm{det}}_n(\mathcal{P})$ do not depend on $\mathcal{R}$. It follows that for each restriction $\mathcal{R}$ and $n, k \in \mathbb{N}_0$

$$e^{\mathrm{ran}}_{n,k}(\mathcal{P}, \mathcal{R}) \le e^{\mathrm{det}}_n(\mathcal{P}).$$

A restricted randomized algorithm is a special case of an (unrestricted) randomized algorithm. Being intuitively clear, this was formally checked in [10], Proposition 2.1 and Corollary 2.2. Moreover, it was shown there that for each restriction $\mathcal{R}$ and $n, k \in \mathbb{N}_0$

$$e_n^{\mathrm{ran}}(\mathcal{P}) \le e_{n,k}^{\mathrm{ran}}(\mathcal{P}, \mathcal{R}),$$

where $e_n^{\mathrm{ran}}(\mathcal{P})$ denotes the n-th minimal error in the randomized setting.

3 Deterministic Versus Restricted Randomized Algorithms

In this section we derive a relation between minimal restricted randomized errors and minimal deterministic errors for general problems. Variants of the following result have been obtained for non-adaptive random bit algorithms in [11, Proposition 1], and for adaptive algorithms that ask for random bits and function values in alternating order in [5]. Obviously, the latter does not permit to analyze a trade-off between the number of random bits and the number of function values to be used in a computation.

Theorem 1 *For all problems $\mathcal{P} = (F, G, S, K, \Lambda)$ and probability spaces with finite access restriction $\mathcal{R} = \big((\Omega, \Sigma, \mathbb{P}), K', \Lambda'\big)$, see (7), and for all $n, k \in \mathbb{N}_0$ we have*

$$e_{n,k}^{\mathrm{ran}}(\mathcal{P}, \mathcal{R}) \ge \frac{1}{3} e_{3n|K'|^{3k}}^{\mathrm{det}}(\mathcal{P}).$$

Without loss of generality in the sequel we only consider access restrictions with the property $K \cap K' = \emptyset$, thus $\bar{K} = K \cup K'$, $\bar{\Lambda} = \Lambda \cup \Lambda'$.

Lemma 1 *Let $n, k \in \mathbb{N}_0$, let A be a randomized algorithm for $\mathcal{P}$ with access restriction $\mathcal{R} = \big((\Omega, \Sigma, \mathbb{P}), K', \Lambda'\big)$. For each $f \in F$ let*

$$B_f = \{\omega \in \Omega \colon \mathrm{card}(A, f, \omega) \le n,\ \mathrm{card}'(A, f, \omega) \le k\}. \tag{11}$$

Then there is an $\mathcal{R}$-restricted randomized algorithm $\tilde{A}$ for $\tilde{\mathcal{P}} = (F, \tilde{G}, \tilde{S}, \Lambda, K)$, where $\tilde{G} = G \oplus \mathbb{R}$ and $\tilde{S} = (S(f), 0)$, satisfying for all $f \in F$ and $\omega \in \Omega$

$$\mathrm{card}(\tilde{A}, f, \omega) \le n \tag{12}$$

$$\mathrm{card}'(\tilde{A}, f, \omega) \le k \tag{13}$$

$$\tilde{A}(f, \omega) = (A(f, \omega) \cdot 1_{B_f}(\omega), 1_{B_f}(\omega)). \tag{14}$$

Proof Let $A = ((L_i)_{i=1}^{\infty}, (\tau_i)_{i=0}^{\infty}, (\varphi_i)_{i=0}^{\infty})$. For $i \in \mathbb{N}_0$ and $a = (a_1, \ldots, a_i) \in \overline{K}^i$ let

$$\begin{aligned} d_{i+1}(a) &= |\{L_1, L_2(a_1), \ldots, L_{i+1}(a_1, \ldots, a_i)\} \cap \Lambda| \\ d'_{i+1}(a) &= \left|\{L_1, L_2(a_1), \ldots, L_{i+1}(a_1, \ldots, a_i)\} \cap \Lambda'\right| \\ \zeta_i(a) &= \begin{cases} 1 \text{ if } (d_{i+1}(a) > n) \vee (d'_{i+1}(a) > k) \\ 0 \quad \text{otherwise.} \end{cases} \end{aligned}$$

Now we define $\tilde{A} = ((L_i)_{i=1}^{\infty}, (\tilde{\tau}_i)_{i=0}^{\infty}, (\tilde{\varphi}_i)_{i=0}^{\infty})$ by setting for $i \in \mathbb{N}_0$ and $a \in \overline{K}^i$

$$\begin{aligned} \tilde{\tau}_i(a) &= \max(\tau_i(a), \zeta_i(a)) \\ \tilde{\varphi}_i(a) &= \begin{cases} (\varphi_i(a), 1) & \text{if } \ \zeta_i(a) \le \tau_i(a) \\ (0, 0) & \text{if } \ \zeta_i(a) > \tau_i(a). \end{cases} \end{aligned}$$

To show (12)–(14) we fix $f \in F$, $\omega \in \Omega$ and define

$$a_1 = L_1(f, \omega), \qquad a_i = (L_i(a_1, \ldots, a_{i-1}))(f, \omega) \quad (i \ge 2).$$

Let $m = \overline{\text{card}}(A, f, \omega)$ and let q be the smallest number $q \in \mathbb{N}_0$ with $\zeta_q(a_1, \ldots, a_q) = 1$. First assume that $\omega \in B_f$. Then for all $i < m$

$$(d_{i+1}(a_1, \ldots, a_i) \le n) \wedge (d'_{i+1}(a_1, \ldots, a_i) \le k),$$

thus $\zeta_i(a_1, \ldots, a_i) = 0$ and therefore $\tilde{\tau}_i(a_1, \ldots, a_i) = 0$. Furthermore,

$$\zeta_i(a_1, \ldots, a_m) \le \tau_m(a_1, \ldots, a_m) = 1,$$

which means $\overline{\text{card}}(\tilde{A}, f, \omega) = m$,

$$\begin{aligned} \text{card}(\tilde{A}, f, \omega) &= d_m(a_1, \ldots, a_{m-1}) \le n \\ \text{card}'(\tilde{A}, f, \omega) &= d'_m(a_1, \ldots, a_{m-1}) \le k \\ \tilde{A}(f, \omega) &= (\varphi_m(a_1, \ldots, a_m), 1) = (A(f, \omega), 1). \end{aligned}$$

Now let $\omega \in \Omega \setminus B_f$, hence

$$\begin{aligned} \tau_0 = \tau_1(a_1) = \cdots = \tau_q(a_1, \ldots, a_q) = 0 \\ (d_{q+1}(a_1, \ldots, a_q) > n) \vee (d'_{q+1}(a_1, \ldots, a_q) > k), \end{aligned}$$

thus $\tilde{\tau}_q(a_1, \ldots, a_q) = 1$. Consequently,

$$\begin{aligned} \text{card}(\tilde{A}, f, \omega) &\le d_q(a_1, \ldots, a_{q-1}) \le n \\ \text{card}'(\tilde{A}, f, \omega) &\le d'_q(a_1, \ldots, a_{q-1}) \le k \\ \tilde{A}(f, \omega) &= (0, 0). \end{aligned}$$

□

The key ingredient of the proof of Theorem 1 is the following.

Lemma 2 *Let $n, k \in \mathbb{N}_0$ and let A be a randomized algorithm for $\mathcal{P}$ with finite access restriction $\mathcal{R} = \big((\Omega, \Sigma, \mathbb{P}), K', \Lambda'\big)$ such that*

$$\operatorname{card}(A, f, \omega) \le n, \quad \operatorname{card}'(A, f, \omega) \le k \tag{15}$$

for all $f \in F$ and $\omega \in \Omega$. Then there exists a deterministic algorithm A^* for $\mathcal{P}$ with

$$A^*(f) = \mathbb{E}\,(A(f, \cdot)), \quad \operatorname{card}(A^*, f) \le n|K'|^k \quad (f \in F). \tag{16}$$

Proof Let $\mathcal{P} = (F, G, S, K, \Lambda), A = ((L_i)_{i=1}^\infty, (\tau_i)_{i=0}^\infty, (\varphi_i)_{i=0}^\infty)$. We argue by induction over $m = n + k$. If $m = 0$, then $\tau_0 = 1$, hence $\operatorname{card}(A, f, \omega) = 0$, thus $A(f, \omega) = \varphi_0$ for all $f \in F$ and $\omega \in \Omega$, and the result follows.

Now let $m \ge 1$. We can assume that $\tau_0 = 0$, otherwise A satisfies (15) with $n = k = 0$ and we are back to the case $m = 0$. Let $\tilde{K} \subset \overline{K}$ be defined by

$$\tilde{K} = \begin{cases} \big\{u \in K : L_1^{-1}(\{u\}) \ne \emptyset\big\} & \text{if} \quad L_1 \in \Lambda \\ \big\{u \in K' : \mathbb{P}(L_1^{-1}(\{u\})) \ne 0\big\} & \text{if} \quad L_1 \in \Lambda'. \end{cases}$$

For every $u \in \tilde{K}$ we define a problem $\mathcal{P}_u = (F_u, G, S_u, K, \Lambda_u)$ and a probability space with access restriction $\mathcal{R}_u = \big((\Omega_u, \Sigma_u, \mathbb{P}_u), K', \Lambda'_u\big)$ as follows. If $L_1 \in \Lambda$, we set $\mathcal{R}_u = \mathcal{R}$ and

$$F_u = \{f \in F : L_1(f) = u\}, \quad S_u = S|_{F_u}, \quad \Lambda_u = \{\lambda|_{F_u} : \lambda \in \Lambda\}.$$

If $L_1 \in \Lambda'$, we put $\mathcal{P}_u = \mathcal{P}$ and

$$\Omega_u = \{\omega \in \Omega : L_1(\omega) = u\}, \quad \Sigma_u = \{B \cap \Omega_u : B \in \Sigma\}$$
$$\mathbb{P}_u(C) = \mathbb{P}\,(\Omega_u)^{-1}\,\mathbb{P}\,(C) \quad (C \in \Sigma_u), \quad \Lambda'_u = \{\lambda'|_{\Omega_u} : \lambda \in \Lambda'\}.$$

Let $\varrho_u : \Lambda \cup \Lambda' \to \Lambda_u \cup \Lambda'_u$ be defined as

$$\varrho_u(\lambda) = \begin{cases} \lambda|_{F_u} & \text{if} \quad \lambda \in \Lambda \\ \lambda|_{\Omega_u} & \text{if} \quad \lambda \in \Lambda' \end{cases}$$

and let $\sigma_u : \Lambda_u \cup \Lambda'_u \to \Lambda \cup \Lambda'$ be any mapping satisfying

$$\varrho_u \circ \sigma_u = \mathrm{id}_{\Lambda_u \cup \Lambda'_u}. \tag{17}$$

Furthermore, we define a random algorithm $A_u = ((L_{i,u})_{i=1}^\infty, (\tau_{i,u})_{i=0}^\infty, (\varphi_{i,u})_{i=0}^\infty)$ for $\mathcal{P}_u$ with access restriction $\mathcal{R}_u$ by setting for $i \ge 0, z_1, \dots, z_i \in \overline{K}$

$$L_{i+1,u}(z_1,\dots,z_i) = \big(\varrho_u \circ L_{i+2}\big)(u,z_1,\dots,z_i) \tag{18}$$
$$\tau_{i,u}(z_1,\dots,z_i) = \tau_{i+1}(u,z_1,\dots,z_i) \tag{19}$$
$$\varphi_{i,u}(z_1,\dots,z_i) = \varphi_{i+1}(u,z_1,\dots,z_i) \tag{20}$$

(in this and similar situations below the case $i = 0$ with variables $z_1,\dots,z_i$ is understood in the obvious way: no dependence on $z_1,\dots,z_i$).

Next we establish the relation of the algorithms A_u to A. Fix $f \in F_u$, $\omega \in \Omega_u$, and let $(a_i)_{i=1}^{\infty} \subseteq \overline{K}$ be given by

$$a_1 = L_1(f,\omega) = u \tag{21}$$
$$a_i = \big(L_i(a_1,\dots,a_{i-1})\big)(f,\omega) \quad (i \geq 2), \tag{22}$$

and similarly $(a_{i,u})_{i=1}^{\infty} \subseteq \overline{K}$ by

$$a_{i,u} = \big(L_{i,u}(a_{1,u},\dots,a_{i-1,u})\big)(f,\omega). \tag{23}$$

We show by induction that

$$a_{i,u} = a_{i+1} \quad (i \in \mathbb{N}). \tag{24}$$

Let $i = 1$. Then (23), (18), (21), and (22) imply

$$a_{1,u} = L_{1,u}(f,\omega) = \big(L_2(u)\big)(f,\omega) = \big(L_2(a_1)\big)(f,\omega) = a_2.$$

For the induction step we let $j \in \mathbb{N}$ and suppose that (24) holds for all $i \leq j$. Then (23), (18), (24), and (22) yield

$$\begin{aligned} a_{j+1,u} &= (L_{j+1,u}(a_{1,u},\dots,a_{j,u}))(f,\omega) = (L_{j+2}(u,a_{1,u},\dots,a_{j,u}))(f,\omega) \\ &= (L_{j+2}(a_1,a_2,\dots,a_{j+1}))(f,\omega) = a_{j+2}. \end{aligned}$$

This proves (24). As a consequence of this relation and of (18), (19), and (20) we obtain for all $i \in \mathbb{N}_0$

$$\begin{aligned} L_{i+1,u}(a_{1,u},\dots,a_{i,u}) &= \big(\varrho_u \circ L_{i+2}\big)(u,a_{1,u},\dots,a_{i,u}) = \big(\varrho_u \circ L_{i+2}\big)(a_1,\dots,a_{i+1}) \\ \tau_{i,u}(a_{1,u},\dots,a_{i,u}) &= \tau_{i+1}(u,a_{1,u},\dots,a_{i,u}) = \tau_{i+1}(a_1,\dots,a_{i+1}) \\ \varphi_{i,u}(a_{1,u},\dots,a_{i,u}) &= \varphi_{i+1}(u,a_{1,u},\dots,a_{i,u}) = \varphi_{i+1}(a_1,\dots,a_{i+1}). \end{aligned}$$

Hence, for all $f \in F_u$ and $\omega \in \Omega_u$

$$\begin{aligned} \overline{\mathrm{card}}(A_u,f,\omega) &= \overline{\mathrm{card}}(A,f,\omega) - 1 \\ A_u(f,\omega) &= A(f,\omega). \end{aligned} \tag{25}$$

Furthermore, if $L_1 \in \Lambda$, then

$$\begin{aligned}\operatorname{card}(A_u, f, \omega) &= \operatorname{card}(A, f, \omega) - 1 \le n - 1\\ \operatorname{card}'(A_u, f, \omega) &= \operatorname{card}'(A, f, \omega) \le k,\end{aligned}$$

and if $L_1 \in \Lambda'$,

$$\begin{aligned}\operatorname{card}(A_u, f, \omega) &= \operatorname{card}(A, f, \omega) \le n\\ \operatorname{card}'(A_u, f, \omega) &= \operatorname{card}'(A, f, \omega) - 1 \le k - 1.\end{aligned}$$

Now we apply the induction assumption and obtain a deterministic algorithm

$$A_u^* = ((L_{i,u}^*)_{i=1}^\infty, (\tau_{i,u}^*)_{i=0}^\infty, (\varphi_{i,u}^*)_{i=0}^\infty)$$

for $\mathcal{P}_u$ with

$$A_u^*(f) = \mathbb{E}_{\mathbb{P}_u}(A_u(f, \cdot)) \tag{26}$$

and

$$\operatorname{card}(A_u^*, f) \le \begin{cases} (n-1)|K'|^k & \text{if} \quad L_1 \in \Lambda \\ n|K'|^{k-1} & \text{if} \quad L_1 \in \Lambda' \end{cases} \tag{27}$$

for every $f \in F_u$.

Finally we use the algorithms A_u^* to compose a deterministic algorithm

$$A^* = ((L_i^*)_{i=1}^\infty, (\tau_i^*)_{i=0}^\infty, (\varphi_i^*)_{i=0}^\infty)$$

for $\mathcal{P}$. This and the completion of the proof is done separately for each of the cases $L_1 \in \Lambda$ and $L_1 \in \Lambda'$.

If $L_1 \in \Lambda$, then we set

$$L_1^* = L_1, \quad \tau_0^* = \tau_0 = 0, \quad \varphi_0^* = \varphi_0,$$

furthermore, for $i \in \mathbb{N}$, $z_1 \in \tilde{K}$, $z_2, \dots, z_i \in \overline{K}$ we let (with σ_{z_1} defined by (17))

$$L_{i+1}^*(z_1, \dots, z_i) = \big(\sigma_{z_1} \circ L_{i,z_1}^*\big)(z_2, \dots, z_i) \tag{28}$$

$$\tau_i^*(z_1, \dots, z_i) = \tau_{i-1,z_1}^*(z_2, \dots, z_i) \tag{29}$$

$$\varphi_i^*(z_1, \dots, z_i) = \varphi_{i-1,z_1}^*(z_2, \dots, z_i). \tag{30}$$

For $i \ge 1$, $z_1 \in \overline{K} \setminus \tilde{K}$, and $z_2, \dots, z_i \in \overline{K}$ we define

$$L_{i+1}^*(z_1, \dots, z_i) = L_1, \quad \tau_i^*(z_1, \dots, z_i) = 1, \quad \varphi_i^*(z_1, \dots, z_i) = \varphi_0.$$

Let $u \in \tilde{K}$ and $f \in F_u$. We show that

$$A^*(f) = A_u^*(f) \tag{31}$$
$$\operatorname{card}(A^*, f) = \operatorname{card}(A_u^*, f) + 1. \tag{32}$$

Let $(b_i)_{i=1}^{\infty} \subseteq \overline{K}$ be given by

$$b_1 = L_1^*(f) = L_1(f) = u \tag{33}$$
$$b_i = \big(L_i^*(b_1, \ldots, b_{i-1})\big)(f) \quad (i \geq 2), \tag{34}$$

and similarly $(b_{i,u})_{i=1}^{\infty} \subseteq \overline{K}$ by

$$b_{i,u} = \big(L_{i,u}^*(b_{1,u}, \ldots, b_{i-1,u})\big)(f). \tag{35}$$

Then

$$b_{i+1} = b_{i,u} \quad (i \in \mathbb{N}). \tag{36}$$

Indeed, for $i = 1$ we conclude from (34), (33), (28), and (35)

$$b_2 = (L_2^*(b_1))(f) = (L_2^*(u))(f) = L_{1,u}^*(f) = b_{1,u}.$$

Now let $j \in \mathbb{N}$ and assume (36) holds for all $i \leq j$. By (34), (33), (28), and (35)

$$\begin{aligned} b_{j+2} &= (L_{j+2}^*(b_1, b_2, \ldots, b_{j+1}))(f) = (L_{j+2}^*(u, b_{1,u}, \ldots, b_{j,u}))(f) \\ &= (L_{j+1,u}^*(b_{1,u}, \ldots, b_{j,u}))(f) = b_{j+1,u}. \end{aligned}$$

This proves (36). It follows from (36), (33), (29), and (30) that for all $i \in \mathbb{N}_0$

$$\begin{aligned} \tau_{i+1}^*(b_1, \ldots, b_{i+1}) &= \tau_{i+1}^*(u, b_{1,u}, \ldots, b_{i,u}) = \tau_{i,u}^*(b_{1,u}, \ldots, b_{i,u}) \\ \varphi_{i+1}^*(b_1, \ldots, b_{i+1}) &= \varphi_{i+1}^*(u, b_{1,u}, \ldots, b_{i,u}) = \varphi_{i,u}^*(b_{1,u}, \ldots, b_{i,u}). \end{aligned}$$

This shows (31) and (32). From (31), (26), and (25) we conclude for $u \in \tilde{K}$, $f \in F_u$, recalling that $\mathcal{R}_u = \mathcal{R}$,

$$A^*(f) = A_u^*(f) = \mathbb{E}_{\mathbb{P}}(A_u(f, \cdot)) = \mathbb{E}_{\mathbb{P}}(A(f, \cdot)).$$

Since $\cup_{u \in \tilde{K}} F_u = F$, the first relation of (16) follows. The second relation is a direct consequence of (32) and (27), completing the induction for the case $L_1 \in \Lambda$.

If $L_1 \in \Lambda'$, then we use the algorithms $(A_u^*)_{u \in \tilde{K}}$ for $\mathcal{P}_u = \mathcal{P}$ and Lemma 3 of [8] to obtain a deterministic algorithm A^* for $\mathcal{P}$ such that for $f \in F$

$$A^*(f) = \sum_{u \in \tilde{K}} \mathbb{P}(L_1^{-1}(\{u\}) A_u^*(f) \tag{37}$$
$$\operatorname{card}(A^*, f) = \sum_{u \in \tilde{K}} \operatorname{card}(A_u^*, f). \tag{38}$$

It follows from (37), (26), and (25) that

$$\begin{aligned} A^*(f) &= \sum_{u\in K':\,\mathbb{P}(L_1^{-1}(\{u\}))>0} \mathbb{P}(L_1^{-1}(\{u\}))\mathbb{E}_{\mathbb{P}_u} A_u(f,\cdot) \\ &= \sum_{u\in K':\,\mathbb{P}(L_1^{-1}(\{u\}))>0} \int_{L_1^{-1}(\{u\})} A_u(f,\omega)d\mathbb{P}(\omega) \\ &= \sum_{u\in K':\,\mathbb{P}(L_1^{-1}(\{u\}))>0} \int_{L_1^{-1}(\{u\})} A(f,\omega)d\mathbb{P}(\omega) = \mathbb{E}_{\mathbb{P}} A_u(f,\cdot). \end{aligned}$$

Furthermore, (27) and (38) imply $\operatorname{card}(A^*, f) \le n|K'|^k$. □

Proof of Theorem 1 The proof is similar to the proof of [5, Lemma 11]. Let $\delta > 0$ and let

$$A = ((L_i)_{i=1}^\infty, (\tau_i)_{i=0}^\infty, (\varphi_i)_{i=0}^\infty) \in \mathcal{A}_{n,k}^{\mathrm{ran}}(\mathcal{P}, \mathcal{R})$$

be a randomized algorithm for $\mathcal{P}$ with restriction $\mathcal{R}$ satisfying

$$e(A, \mathcal{P}) \le e_{n,k}^{\mathrm{ran}}(\mathcal{P}, \mathcal{R}) + \delta. \tag{39}$$

For $f \in F$ define

$$B_f = \{\omega \in \Omega\colon \operatorname{card}(A, f, \omega) \le 3n,\ \operatorname{card}'(A, f, \omega) \le 3k\}.$$

Observe that $B_f \in \Sigma$ and $P(B_f) \ge 1/3$. For the conditional expectation

$$\mathbb{E}\,(A(f,\cdot) \mid B_f) = \frac{\mathbb{E}\,\left(A(f,\cdot)\cdot 1_{B_f}\right)}{P(B_f)}$$

of $A(f,\cdot)$ given B_f we obtain

$$\begin{aligned} &3\mathbb{E}\,\|S(f) - A(f,\cdot)\|_G \\ &\ge \mathbb{E}\,\left(\|S(f) - A(f,\cdot)\|_G \mid B_f\right) \ge \left\|S(f) - \mathbb{E}\,\left(A(f,\cdot)\mid B_f\right)\right\|_G \end{aligned} \tag{40}$$

by means of Jensen's inequality. Our goal is now to design a deterministic algorithm with input-output mapping $f \mapsto \mathbb{E}\,(A(f,\cdot) \mid B_f)$.

From Lemma 1 we conclude that there is an $\mathcal{R}$-restricted randomized algorithm $\tilde{A} = ((L_i)_{i=1}^\infty, (\tilde{\tau}_i)_{i=0}^\infty, (\tilde{\varphi}_i)_{i=0}^\infty)$ for $\tilde{\mathcal{P}} = (F, \tilde{G}, \tilde{S}, \Lambda, K)$, where $\tilde{G} = G \oplus \mathbb{R}$ and $\tilde{S}(f) = (S(f), 0)$ $(f \in F)$, satisfying for all $f \in F$ and $\omega \in \Omega$

$$\begin{aligned} &\operatorname{card}(\tilde{A}, f, \omega) \le 3n, \quad \operatorname{card}'(\tilde{A}, f, \omega) \le 3k, \\ &\tilde{A}(f,\omega) = (A(f,\omega)\cdot 1_{B_f}(\omega), 1_{B_f}(\omega)). \end{aligned}$$

By Lemma 2 there is a deterministic algorithm $A^* = ((L_i^*)_{i=1}^\infty, (\tau_i^*)_{i=0}^\infty, (\varphi_i^*)_{i=0}^\infty)$ for $\tilde{\mathcal{P}}$ such that for all $f \in F$

$$\operatorname{card}(A^*, f) \le 3n|K'|^{3k}, \quad A^*(f) = \left(\int_{B_f} A(f,\omega) d\mathbb{P}(\omega), \mathbb{P}(B_f) \right).$$

It remains to modify A^* as follows

$$\tilde{A}^* = ((L_i^*)_{i=1}^\infty, (\tau_i^*)_{i=0}^\infty, (\psi_i^*)_{i=0}^\infty),$$

where for $i \in \mathbb{N}_0$ and $a \in K^i$

$$\psi_i^*(a) = \begin{cases} \frac{\varphi_{i,1}^*(a)}{\varphi_{i,2}^*(a)} & \text{if} \quad \varphi_{i,2}^*(a) \neq 0 \\ 0 & \text{if} \quad \varphi_{i,2}^*(a) = 0, \end{cases}$$

with $\varphi_i^*(a) = (\varphi_{i,1}^*(a), \varphi_{i,2}^*(a))$ being the splitting into the G and $\mathbb{R}$ component. Hence for each $f \in F$

$$\begin{aligned} \operatorname{card}(\tilde{A}^*, f) &\le 3n|K'|^{3k} \\ \tilde{A}^*(f) &= \mathbb{E}\,(A(f,\cdot) \mid B_f), \end{aligned}$$

and therefore we conclude, using (39) and (40),

$$e^{\det}_{3n|K'|^{3k}}(\mathcal{P}) \le e(\tilde{A}^*, \tilde{\mathcal{P}}) \le 3e(A, \mathcal{P}) \le 3(e^{\mathrm{ran}}_{n,k}(\mathcal{P}, \mathcal{R}) + \delta)$$

for each $\delta > 0$. □

4 Applications

4.1 Integration of Functions in Sobolev Spaces

Let $r, d \in \mathbb{N}, 1 \le p < \infty, Q = [0,1]^d$, let $C(Q)$ be the space of continuous functions on Q, and $W_p^r(Q)$ the Sobolev space, see [1]. Then $W_p^r(Q)$ is embedded into $C(Q)$ iff

$$(p = 1 \text{ and } \quad r/d \ge 1) \quad \text{or} \quad (1 < p < \infty \text{ and } \quad r/d > 1/p). \tag{41}$$

Let $B_{W_p^r(Q)}$ be the unit ball of $W_p^r(Q)$, $B_{W_p^r(Q)} \cap C(Q)$ the set of those elements of the unit ball which are continuous (more precisely, of equivalence classes, which contain a continuous representative), and define

$$F_1 = \begin{cases} B_{W_p^r(Q)} & \text{if the embedding condition (41) holds} \\ B_{W_p^r(Q)} \cap C(Q) & \text{otherwise.} \end{cases}$$

Moreover, let $I_1 : W_p^r(Q) \to \mathbb{R}$ be the integration operator

$$I_1 f = \int_Q f(x)dx.$$

and let $\Lambda_1 = \{\delta_x : x \in Q\}$ be the set of point evaluations, where $\delta_x(f) = f(x)$. Put into the general framework of (1), we consider the problem $\mathcal{P}_1 = (F_1, \mathbb{R}, I_1, \mathbb{R}, \Lambda_1)$. Set $\bar{p} = \min(p, 2)$. Then the following is known (for (42–44) below see [9] and references therein). There are constants $c_{1-6} > 0$ such that for all $n \in \mathbb{N}_0$

$$c_1 n^{-r/d-1+1/\bar{p}} \le e_n^{\mathrm{ran}}(\mathcal{P}_1) \le c_2 n^{-r/d-1+1/\bar{p}}, \tag{42}$$

moreover, if the embedding condition holds, then

$$c_3 n^{-r/d} \le e_n^{\mathrm{det}}(\mathcal{P}_1) \le c_4 n^{-r/d}, \tag{43}$$

while if the embedding condition does not hold, then

$$c_5 \le e_n^{\mathrm{det}}(\mathcal{P}_1) \le c_6. \tag{44}$$

Theorem 1 immediately gives (compare this with the rate in the unrestricted setting (42)).

Corollary 1 *Assume that the embedding condition (41) does not hold and let $\mathcal{R}$ be any finite access restriction, see (7). Then there is a constant $c > 0$ such that for all $n, k \in \mathbb{N}$*

$$e_{n,k}^{\mathrm{ran}}(\mathcal{P}_1, \mathcal{R}) \ge c.$$

It was shown in [11], that if the embedding condition holds, then $(2 + d) \log_2 n$ random bits suffice to reach the rate of the unrestricted randomized setting, thus, if $\mathcal{R}$ is a bit restriction (see (8)–(9)), then there are constants $c_1, c_2 > 0$ such that for all $n \in \mathbb{N}$

$$c_1 n^{-r/d-1+1/\bar{p}} \le e_n^{\mathrm{ran}}(\mathcal{P}_1) \le e_{n,(2+d)\log_2 n}^{\mathrm{ran}}(\mathcal{P}_1, \mathcal{R}) \le c_2 n^{-r/d-1+1/\bar{p}}. \tag{45}$$

The following consequence of Theorem 1 shows that the number of random bits used in the (non-adaptive) algorithm from [11] giving (45) is optimal up to a constant factor, also for adaptive algorithms.

Corollary 2 *Assume that the embedding condition holds and let $\mathcal{R}$ be any finite access restriction. Then for each σ with $0 < \sigma \le 1 - 1/\bar{p}$ and each $c_0 > 0$ there are constants $c_1 > 0$, $c_2 \in \mathbb{R}$ such that for all $n, k \in \mathbb{N}$*

$$e_{n,k}^{\mathrm{ran}}(\mathcal{P}_1, \mathcal{R}) \le c_0 n^{-r/d-\sigma}.$$

implies

$$k \ge c_1 \sigma \log_2 n + c_2.$$

Proof Let $\mathcal{R} = \big((\Omega, \Sigma, \mathbb{P}), K', \Lambda'\big)$. By Theorem 1 and (43),

$$c_0 n^{-r/d-\sigma} \ge e_{n,k}^{\mathrm{ran}}(\mathcal{P}_1, \mathcal{R}) \ge 3^{-1} e_{3n|K'|^{3k}}^{\mathrm{det}}(\mathcal{P}_1) \ge 3^{-1} c_3 (n|K'|^{3k})^{-r/d},$$

implying

$$\log_2 c_0 - \sigma \log_2 n \ge \log_2(c_3/3) - \frac{3kr}{d} \log_2 |K'|,$$

thus,

$$k \ge \frac{d}{3r \log_2 |K'|} (\sigma \log_2 n - \log_2 c_0 + \log_2(c_3/3)).$$

□

4.2 Integration of Lipschitz Functions Over the Wiener Space

Let μ be the Wiener measure on $C([0, 1])$,

$$F_2 = \{f : C([0, 1]) \to \mathbb{R},\ |f(x) - f(y)| \le \|x - y\|_{C([0,1])} \quad (x, y \in C([0, 1]))\},$$

$G = \mathbb{R}$, let $I_2 : F \to \mathbb{R}$ be the integration operator given by

$$I_2 f = \int_{C([0,1])} f(x) d\mu(x),$$

and $\Lambda_2 = \{\delta_x : x \in C([0, 1])\}$, so we consider the problem $\mathcal{P}_2 = (F_2, \mathbb{R}, I_2, \mathbb{R}, \Lambda_2)$. There exist constants $c_{1-4} > 0$ such that

$$c_1 n^{-1/2} (\log_2 n)^{-3/2} \le e_n^{\mathrm{ran}}(\mathcal{P}_2) \le c_2 n^{-1/2} (\log_2 n)^{-1/2} \tag{46}$$

and

$$c_3 (\log_2 n)^{-1/2} \le e_n^{\mathrm{det}}(\mathcal{P}_2) \le c_4 (\log_2 n)^{-1/2} \tag{47}$$

for every $n \ge 2$, see [2], Theorem 1 and Proposition 3 for (47) and Theorems 11 and 12 for (46). Moreover, it is shown in [5], Theorem 8 and Remark 9, that if $\mathcal{R}$ is a bit

restriction, then there exist a constants $c_1 > 0$, $c_2 \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ with $n \geq 3$

$$e^{\mathrm{ran}}_{n,\kappa(n)}(\mathcal{P}_2, \mathcal{R}) \leq c_1 n^{-1/2} (\log_2 n)^{3/2}, \tag{48}$$

where

$$\kappa(n) = c_2 \lceil n (\log_2 n)^{-1} \log_2(\log_2 n) \rceil. \tag{49}$$

Our results imply that the number of random bits (49) used in the algorithm of [5] giving the upper bound in (48) is optimal (up to log terms) in the following sense.

Corollary 3 *Let $\mathcal{R}$ be a finite access restriction. For each $\alpha \in \mathbb{R}$ and each $c_0 > 0$ there are constants $c_1 > 0$ and $c_2 \in \mathbb{R}$ such that for all $n, k \in \mathbb{N}$ with $n \geq 2$*

$$e^{\mathrm{ran}}_{n,k}(\mathcal{P}_2, \mathcal{R}) \leq c_0 n^{-1/2} (\log_2 n)^{\alpha}.$$

implies

$$k \geq c_1 n (\log_2 n)^{-2\alpha} + c_2. \tag{50}$$

Proof Let $\mathcal{R} = \big((\Omega, \Sigma, \mathbb{P}), K', \Lambda'\big)$. We use Theorem 1 again. From (47) we obtain

$$c_0 n^{-1/2} (\log_2 n)^{\alpha} \geq e^{\mathrm{ran}}_{n,k}(\mathcal{P}_2, \mathcal{R}) \geq 3^{-1} e^{\mathrm{det}}_{3n|K'|^{3k}}(\mathcal{P}_2) \geq 3^{-1} c_3 \log_2(3n|K'|^{3k})^{-1/2},$$

thus

$$\log_2(3n) + 3k \log_2 |K'| \geq \frac{c_3^2}{9c_0^2} n (\log_2 n)^{-2\alpha},$$

which implies

$$k \geq (3 \log_2 |K'|)^{-1} \left(\frac{c_3^2}{9c_0^2} n (\log_2 n)^{-2\alpha} - \log_2(3n) \right). \tag{51}$$

Choosing $n_0 \in \mathbb{N}$ in such a way that for $n \geq n_0$

$$\frac{c_3^2}{18c_0^2} n (\log_2 n)^{-2\alpha} \geq \log_2(3n)$$

leads to

$$k \geq (3 \log_2 |K'|)^{-1} \left(\frac{c_3^2}{18c_0^2} n (\log_2 n)^{-2\alpha} - \log_2(3n_0) \right).$$

Acknowledgements I thank Mario Hefter and Klaus Ritter for discussions on the subject of this paper. Furthermore, I am grateful to two anonymous referees for suggestions improving the presentation, and to one of them for coining the name 'randomization functional'.

References

1. Adams, R.A.: Sobolev Spaces. Academic, New York (1975)
2. Creutzig, J., Dereich, S., Müller-Gronbach, Th., Ritter, K.: Infinite-dimensional quadrature and approximation of distributions. Found. Comput. Math. **9**(4), 391–429 (2009)
3. Gao, W., Ye, P., Wang, H.: Optimal error bound of restricted Monte Carlo integration on anisotropic Sobolev classes. Progr. Natur. Sci. (English Ed.) **16**, 588–593 (2006)
4. Giles, M.B., Hefter, M., Mayer, L., Ritter, K.: Random bit quadrature and approximation of distributions on Hilbert spaces. Found. Comput. Math. **19**, 205–238 (2019)
5. Giles, M.B., Hefter, M., Mayer, L., Ritter, K.: Random bit multilevel algorithms for stochastic differential equations. J. Complex. **54**, 101395 (2019)
6. Giles, M.B., Hefter, M., Mayer, L., Ritter, K.: An Adaptive Random Bit Multilevel Algorithm for SDEs. In: Hickernell, F., Kritzer, P. (eds.) Multivariate Algorithms and Information-Based Complexity, pp. 15–32. De Gruyter, Berlin/Boston (2020)
7. Heinrich, S.: Monte Carlo approximation of weakly singular integral operators. J. Complex. **22**, 192–219 (2006)
8. Heinrich, S.: The randomized information complexity of elliptic PDE. J. Complex. **22**, 220–249 (2006)
9. S. Heinrich, Stochastic approximation and applications, In: Plaskota, L., Woźniakowski, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2010, pp. 95–131. Springer, Berlin (2012)
10. Heinrich, S.: On the power of restricted Monte Carlo algorithms. Matrix Ann. **2018**, 45–59 (2020). Springer
11. Heinrich, S., Novak, E., Pfeiffer, H.: How many random bits do we need for Monte Carlo integration? In: Niederreiter, H. (ed.) Monte Carlo and Quasi-Monte Carlo Methods 2002, pp. 27–49. Springer, Berlin (2004)
12. Novak, E.: Monte Carlo-Verfahren zur numerischen Integration. In: Grossmann, W. et al. (eds.) Proceedings of 4th Pannonian Symposium on Mathematical Statistics, pp. 269–282. Bad Tatzmannsdorf, Austria 1983, Reidel (1985)
13. Novak, E.: Deterministic and Stochastic Error Bounds in Numerical Analysis, Lecture Notes in Mathematics, vol. 1349. Springer, Berlin (1988)
14. Novak, E., Pfeiffer, H.: Coin tossing algorithms for integral equations and tractability. Monte Carlo Methods Appl. **10**, 491–498 (2004)
15. Traub, J.F., Wasilkowski, G.W., Woźniakowski, H.: Information-Based Complexity. Academic (1988)
16. Traub, J.F., Woźniakowski, H.: The Monte Carlo algorithm with a pseudorandom generator. Math. Comput. **58**, 323–339 (1992)
17. Ye, P., Hu, X.: Optimal integration error on anisotropic classes for restricted Monte Carlo and quantum algorithms. J. Approx. Theory **150**, 24–47 (2008)

Massively Parallel Path Space Filtering

Nikolaus Binder, Sascha Fricke, and Alexander Keller

Abstract Restricting path tracing to a small number of paths per pixel in order to render images faster rarely achieves a satisfactory image quality for scenes of interest. However, path space filtering may dramatically improve the visual quality by sharing information across vertices of paths classified as proximate. Unlike screen space approaches, these paths neither need to be present on the screen, nor is filtering restricted to the first intersection with the scene. While searching proximate vertices had been more expensive than filtering in screen space, we greatly improve over this performance penalty by storing, updating, and looking up the required information in a hash table. The keys are constructed from jittered and quantized information, such that only a single query very likely replaces costly neighborhood searches. A massively parallel implementation of the algorithm is demonstrated on a graphics processing unit (GPU).

Keywords Integral equations · Real-time light transport simulation · Variance reduction · Hashing · Monte Carlo integration · Massively parallel algorithms

1 Introduction

Realistic image synthesis consists of high-dimensional numerical integration of functions with potentially high variance. Restricting the number of samples therefore often results in visible noise, which efficiently can be reduced by path space filtering [19] as shown in Fig. 1.

N. Binder · A. Keller (✉)
NVIDIA, Fasanenstr. 81, 10623 Berlin, Germany
e-mail: akeller@nvidia.com

N. Binder
e-mail: nbinder@nvidia.com

S. Fricke
University of Braunschweig, 38106 Braunschweig, Germany

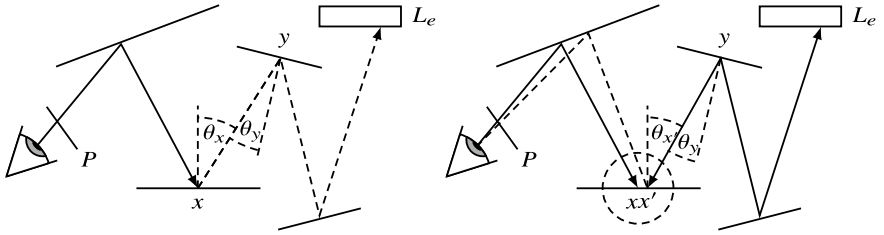
A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_7

Fig. 1 Path tracing at one path per pixel (top) in combination with hashed path space filtering (middle) very closely approximates the reference solution using 1024 paths per pixel (bottom) and does so with an overhead of about 1.5 ms in HD resolution. Scene courtesy of Epic Games

We improve the performance of path space filtering by replacing costly neighborhood search with averages of clusters in voxels resulting from quantization. Our new algorithm is suitable for interactive and even real-time rendering and it enables many applications trading a controllable bias for a dramatic speedup and noise reduction.

2 Light Transport Simulation

As illustrated in Fig. 2, light transport is simulated by tracing rays to create paths that connect the light sources and the camera sensor through a three-dimensional scene that is represented by surfaces and scattering properties of the materials and volumes



(a) forward path tracing, subpath connections, and next event estimation

(b) density estimation and path space filtering

Fig. 2 Geometry of light transport simulation by path tracing. Starting from the eye on the left through the image plane P, a light transport path segment ends in x. (**a**) Forward path tracing continues the path until it terminates on the surface of a light source. A path can also be completed by directly connecting to a point y on the surface of a light source (next event estimation) or to a vertex y of the same or a different path (subpath connection). (**b**) Radiance in point x can directly be evaluated by accumulating radiance in vertices x' in a local neighborhood either for density estimation or path space filtering

of participating media. Light transport is ruled by an integral equation: the incident radiance

$$L_i(x,\omega) = L_e(x,\omega) + L_r(x,\omega) \tag{1}$$

is the sum of the emitted radiance L_e and the reflected radiance L_r in direction ω. The following equations show four equivalent ways to formulate the reflected radiance L_r in a point x in direction ω_r and Fig. 2 illustrates the corresponding sampling techniques:

$$L_r(x,\omega_r)$$

$$= \int_{\mathcal{S}^2_-(x)} L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x d\omega \tag{2}$$

$$= \int_{\partial V} L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \frac{\cos\theta_y}{|x-y|^2} V(x,y) dy \tag{3}$$

$$= \lim_{r(x)\to 0} \int_{\partial V} \int_{\mathcal{S}^2_-(y)} \frac{1_{B(r(x))}\big(x,h(y,\omega)\big)}{\pi r(x)^2} L_i\big(h(y,\omega),\omega\big)\cdot$$

$$\cdot\, f_r\big(\omega_r,h(y,\omega),\omega\big)\cos\theta_y d\omega dy \tag{4}$$

$$= \lim_{r(x)\to 0} \int_{\mathcal{S}^2_-(x)} \frac{\int_{\partial V} 1_{B(r(x))}\big(x,x'\big) w(x,x') L_i(x',\omega) f_r(\omega_r,x,\omega)\cos\theta_{x'} dx'}{\int_{\partial V} 1_{B(r(x))}\big(x,x',r(x)\big) w(x,x') dx'} d\omega \tag{5}$$

Here, the ray tracing function $h(y, \omega)$ determines the closest point of intersection of the scene surface and a ray starting in the point y traced into direction ω. The characteristic function

$$1_{B(r)}(c, x) := \begin{cases} 1 & \|c - x\|_2 < r \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

determines whether the point x is inside the ball B with radius r centered at c or, in other words, whether the points c and x are no further apart than the distance r. The sampling techniques are:

Forward Path Tracing: While in reality paths of photons start on emissive surfaces, and the camera sensor measures the ones terminating on its surface, simulations often construct paths backwards. Somewhat counter-intuitively, such a simulation in reverse photon direction is called "forward path tracing"—forward in view direction. Equation (2) integrates radiance over the upper hemisphere S^2_- by multiplying the incident radiance L_i from the angle ω in the point x with the spatio-directional reflectivity f_r for the two angles in the point x and the cosine between the normal of the surface and the incident ray to account for the change of area of the projected solid angle. Inserting the equation into Eq. (1) and the result back into Eq. (2) allows for subsequently prolonging paths and is known as *(recursive) forward path tracing*.

Next Event Estimation and Subpath Connection: Equation (3) changes the integration domain to the scene surface ∂V so that a path can be constructed in which the point x connects to any point y on the scene surfaces. The integrand is then extended by the mutual visibility V of the two points x and y. The fraction of the cosine of the second angle and the squared distance of the two points accounts for the change of measure. One often refers to this fraction as the *geometric term*. Equation (3) is especially useful since it allows to directly connect to the surface of a light source (*next event estimation*) or any other vertex of any path (*subpath connection*).

Density Estimation: Equation (4) again integrates over the scene surface. However, it realizes density estimation by restricting to a local neighborhood in a sphere with radius $r(x)$ using the characteristic function of the ball 1_B. The density is then obtained by dividing by the area of the circle that stems from the intersection of the ball and the flat surface. For a radius going to zero, the formulation is equivalent to the previous formulations. Density estimation tracing photons from the light sources is referred to as *photon mapping* [17].

Path Space Filtering: Similarly, Eq. (5) integrates over a local neighborhood of x. In contrast to density estimation, a local weighted average is calculated. This local average is normalized by the integral of all weights in the neighborhood instead of the area of a circle. For a finite non-zero radius, the method trades a certain bias for variance reduction. Due to the filtering of the local average this technique has been introduced as *path space filtering* [19].

While implementations of equations (2)–(5) each individually come with their own strengths and weaknesses and are therefore often combined for robustness in offline rendering applications [12, 15, 21, 35], time constraints of real-time image synthesis as well as advances in *path guiding* [6, 26, 27] lead to the vast majority of implementations only employing equations (2) and (3). Our work aims at a substantial acceleration of path space filtering, resulting in a considerable variance reduction in real-time applications at the cost of a controllable amount of bias.

2.1 Previous Work

Filtering results of light transport simulation is gaining more and more attention in real-time, interactive, and even offline rendering. The surveys by Zwicker et al. [37] and Sen et al. [33] present an overview of recent developments. The fastest approaches use only information available at primary intersections and perform filtering in screen space. Further recent work is based on deep neural networks [1, 4], hierarchical filtering with weights based on estimated variance in screen space [31], or on improving performance by simplifying the overall procedure [24].

Fast filtering is also possible in texture space [28], which requires a bijection between the scene surface and texture space. While this may be tricky already, issues may arise along discontinuities of a parametrization in addition. Furthermore, filtering is restricted to locations on surfaces when operating in texture space, and thus volumetric effects must be filtered separately.

Path space filtering [19], on the other hand, averages contributions of light transport paths in path space, which allows for filtering at non-primary intersections and for a more efficient handling of dis-occlusions during temporal filtering. Multiple Importance Sampling weights, for example those for path space filtering, can be further optimized [36]. However, querying the contributions in path space so far had been significantly more expensive than filtering the contributions of neighboring pixels in screen space. Neglecting the fact that locations that are close in path space are not necessarily adjacent in screen space enables interactive filtering in screen space [11]. As a consequence, filtering in screen space is almost only efficient for primary rays or reflections from sufficiently smooth and flat surfaces. In fact, such filtering algorithms are a variant of a bilateral filtering using path space proximity to determine weights. Sharing information across pixels according to a similarity measure dates back as early as the 1990s [18]. Since then, several variants have been introduced, for example by re-using paths in nearby pixels [2], for filtering by anisotropic diffusion [25] or using edge-avoiding Á-Trous wavelets [7].

Kontkanen et al. explore irradiance filtering, a subset of path space filtering [20]. Spatial caching of shading results in a hash table for walkthroughs of static scenes [8] uses similar methods to the ones presented in this work for the lookup of these results. Again, the method can be seen as a subset of path space filtering: It is restricted to caching diffuse illumination and neither includes filtering nor spatial and temporal integration in an arbitrary number of vertices of a light path.

Hachisuka et al. also use a hash table in a light transport simulation on the GPU [14]. The approach is fundamentally different in two aspects: First, it traces photons and stores them in the hash table for density estimation, while our method averages radiance in vertices from arbitrary light paths. Second, their method implements simple sampling without replacement in voxels, culling all but one photon per voxel. Our method does the exact opposite: It collects radiance from all paths whose vertices coincide in a voxel.

Mara et al. summarize and evaluate a number of methods for photon mapping on the GPU [23]. Their evaluation also includes work from Ma and McCool using hash tables with lists of photons in per voxel [22]. While all examined methods may be used for path space filtering instead of photon mapping, their performance is at least limited by the maintenance of lists.

Havran et al. use two trees for final gathering with photon mapping [16]. The overhead of tree construction and traversal as well as iterating through lists of vertices severely limit the performance in our intended real-time use case.

3 Algorithm

Like path space filtering [19], the algorithm receives a set of vertices in which radiance should be filtered to reduce variance. For each vertex complementary information such as the surface normal, the attenuation from the camera along the light transport path up to the vertex, and incident radiance is provided.

A first phase generates the aforementioned data from light transport paths, for example, by path tracing. The second phase averages the radiance of vertices in voxels, as described in Sect. 3.1, and stores and looks up the averages in a hash table, see Sect. 3.2. Finally, for each vertex its associated average is multiplied by its attenuation and accumulated in its respective pixel. Techniques described in Sect. 3.3 reduce the variance of voxels with a small number of vertices. Sect. 3.4 discusses filtering over time for interactive light transport simulation.

3.1 Averaging in Voxels

Rather than averaging the radiance of vertices in a three-dimensional ball, we partition the space of vertex information into voxels and compute one average per voxel. We therefore introduce the concept of a key k of a vertex x. The key is a subset of the data stored for a vertex and at least contains the 3-dimensional position of the vertex x along with possibly other vertex data, for example, the surface normal (see Sect. 3.1.1). The voxels result from quantizing the scaled components of a key vector to integers. The scale defines the size of the voxels and is determined by the resolution selection function $s(k)$, which itself may depend on the key (see

Sect. 3.1.2). It balances bias and variance, while remaining quantization artifacts are taken care of by stochastic interpolation (see Sect. 3.1.3).

We define the characteristic function

$$1_V(k,k') := \begin{cases} 1 & \lfloor s(k) \odot k \rfloor = \lfloor s(k') \odot k' \rfloor \wedge s(k) = s(k') \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

telling us whether two key vectors k and k' of vertices x and x', respectively, share the same voxel. Note that sharing the same voxel requires identical scale vectors, too. Here $\odot$ denotes component-wise multiplication and the floor function $\lfloor \cdot \rfloor$ is applied per component. Depending on the number of components selected for a key, voxels are not necessarily three-dimensional, and their extent may vary between components.

Replacing the characteristic function 1_B by 1_V in path space filtering as given by equation (5), selecting the weight $w(x,x') \equiv 1$, and considering an approximation rather than the limit, we obtain

$$L_r(x,\omega_r) \approx \int_{\mathcal{S}^2_\perp(x)} \frac{\int_{\partial V} 1_V(k,k') L_i(x',\omega) f_r(\omega_r,x,\omega) \cos\theta_{x'} dx'}{\int_{\partial V} 1_V(k,k') dx'} d\omega. \tag{8}$$

If $f_r(\omega_r,x,\omega)$ is separable into $f_r(\omega_r,x) \cdot f_i(x,\omega)$, and $f_i(x,\omega)$ is—at least approximately—constant within the voxel, we will be able to rewrite Eq. (8) as

$$L_r(x,\omega_r) \approx f_r(\omega_r,x) \int_{\mathcal{S}^2_\perp(x)} \frac{\int_{\partial V} 1_V(k,k') L_i(x',\omega) f_i(x',\omega) \cos\theta_{x'} dx'}{\int_{\partial V} 1_V(k,k') dx'} d\omega. \tag{9}$$

In the following, we call the product $L_i(x',\omega) \cdot f_i(x',\omega) \cdot \cos\theta_{x'}$ the *contribution* of the vertex in x'. Since all terms of the integrand except for $1_V(k,k')$ are independent of x and ω_r, it is now possible to calculate the integral in Eq. (9) only once for all vertices sharing a voxel. Thereby, the integral calculated per voxel is independent of those components excluded from the key.

3.1.1 Construction of Keys

The key k of a vertex is a vector that contains a subset of the information stored with a vertex x. This vector incorporates all information required to cluster proximate vertices. The selection of components is critically important for defining the tradeoff between bias and variance reduction: including additional components of the vertex information in the key may reduce the bias, while excluding components allows for the inclusion of more vertices in the integral in Eq. (9), therefore reducing variance. In the following we will give an overview of components (see Fig. 3) that one would typically include in the key.

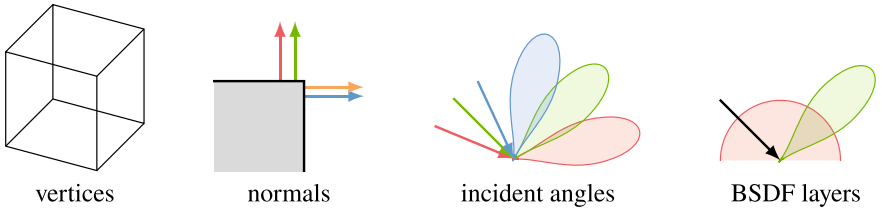


Fig. 3 The components of the key vector k of a vertex usually consist of the coordinates of the point x of the vertex and optionally may include the normal in x, the angle of incidence of radiance, and an identifier of a layer of a bidirectional scattering distribution function (BSDF)

The quality of the approximation in Eq. (9) highly depends on the deviation of L_i in x' from the one in x. First and foremost, it is therefore recommended to restrict the world space extent of the voxel by including the position x of the vertex in the key k. While L_i is not continuous in practice—for example along shadow edges—the visible error of the approximation decreases with the world space extent of a voxel.

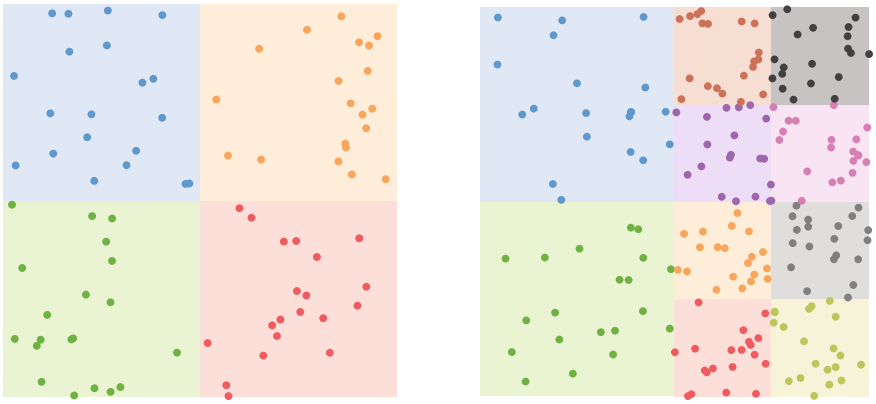
In practice, one can furthermore not guarantee that $\lim_{x' \to x} \cos \theta_{x'} = \cos \theta_x$ due to different surface orientations in the two locations, for example along edges of objects. Including the normal of the surface in the point x in the key avoids a potential "smearing" around edges and "flattening" of surfaces. Representations of unit vectors are surveyed in [5].

Splitting $f_r(\omega_r, x, \omega)$ into $f_r(\omega_r, x) \cdot f_i(x, \omega)$ is not always possible. On highly reflective surfaces, f_r is defined as a Dirac delta function, and filtering is pointless. Therefore, vertices on such surfaces should not be selected in the first place. On glossy surfaces, however, filtering may reduce variance efficiently, again at the cost of a certain bias. While f_r cannot be split on these surfaces without unpleasantly and undesirably changing the visual appearance, partitioning the domain of incident angles and computing separate averages for each interval may be a viable tradeoff. Therefore, for vertices on such surfaces one can append the incident angle to the key in order to identify vertices with similar incident angle.

Materials are often composed of different layers with different properties. Filtering the layers independently offers the opportunity to use different voxel resolutions as well as constructing keys with different components for the different layers. For example, a material consisting of a glossy layer on top of a diffuse layer could only include the angle ω_r in the key used for filtering the glossy layer since the attenuation of the diffuse layer is independent of it. In turn, the integral for the diffuse layer benefits from including more samples. Appending an identifier of the layer to the key partitions the average into several individual ones, which can be combined later.

3.1.2 Adaptive Resolution

In the simplest case, the resolution selection function $s(k)$ is a constant. In practice, it is often advisable to increase the world space extent of a voxel with its distance to the



(a) $s(k) \propto 1$ (b) $s(k) \propto 2^{\lfloor \log_2 x_0 \rfloor}$

Fig. 4 The characteristic function of a voxel $1_V(k, k')$ is 1 for all k, k' inside the same voxel, here depicted by the same color. A constant resolution selection function $s(k)$ yields uniformly sized voxels (**a**), while increasing $s(k)$ along the x_0-axis from left to right shrinks voxels according to distance (**b**) reminiscent of a quad-tree structure

camera sensor along the light transport path. Then, one can filter more aggressively in distant voxels, and increasing the size counteracts the decrease of the density of vertices from paths directly coming from the camera sensor with increasing distance. Our implementation parameterizes $s(k)$ by defining an area on the screen, and then calculates the projected size of the area on the screen using the projection theorem. Fig. 4 illustrates the principle for sets of two-dimensional keys defined by this characteristic function. In practice, keys are at least three-dimensional, i.e. defined by the world space position of the vertex.

The choice of the resolution selection function $s(k)$ is crucial for finding a good tradeoff between visible bias and reduction of variance. Shrinking the voxels by increasing $s(k)$ reduces bias by averaging over a smaller neighborhood, but increases variance. In theory, $s(k)$ should be large in areas with a lot of high frequency detail in L_i. In practice, those areas are almost always unknown since L_i is unknown. Fig. 5 shows how a sharp shadow is blurred due to averaging radiance in a large voxel. One would therefore like to adaptively chose a finer resolution along its boundary.

Finite spatial differences may be used in heuristics for adaptation. While their computation either introduces a certain overhead or reduces the number of independent samples, cost may be amortized over frames in environments changing only slowly over time. Note that finite differences only estimate spatial variations of the averages, and one must therefore carefully both choose and adjust such heuristics as well as determine the number samples used for finite spatial differences.

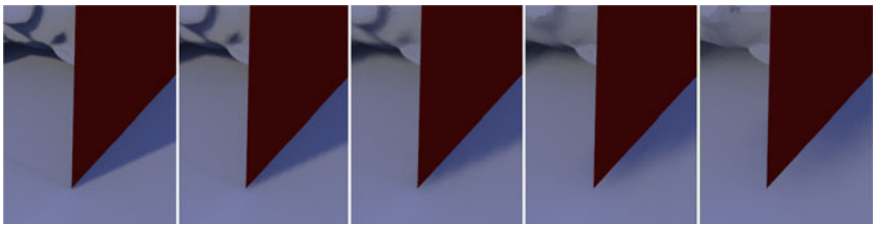


Fig. 5 Increasing the filter size by lowering the resolution $s(k)$ (left to right) increasingly blurs shadows and also increases the amount of light and shadow leaking. The leftmost image is representative of the reference solution

3.1.3 Filter Kernel Approximation by Jittering

The discontinuities of quantization are removed by jittering components of the key, which in fact amounts to approximating a filter kernel by sampling. Jittering depends on the kind of component of the key, for example, positions are jittered in the tangent plane of an intersection, see Algorithm 1. The resulting noise is clearly preferable over the visible discretization artifacts resulting from quantization, as illustrated and shown in Figs. 6 and 7. In contrast to discretization artifacts, noise from jitter is less perceptually pronounced and simple to filter. Note that jittering is not limited to the position; in fact jittering is advantageous for all continuous components of the key that may suffer discontinuities introduced by quantization, such as the normal.

Jittering can be performed either before accumulation or before lookup, leading to similar results. In practice, we suggest using the same jittered key for both accumulation and lookup. Then, the contribution is guaranteed to be included in the average. Furthermore, this allows us to determine the index in the hash table only once for accumulation, and re-use the index later for the lookup of the average without having to calculate the two hash functions and solve potential collisions with linear probing redundantly.

Fig. 6 Jittering trades quantization artifacts for noise. Left: Note that the resolution $s(k)$ at the jittered location (red) may differ from the one of the original location (green). Spatial jittering hides otherwise visible quantization artifacts (middle): The resulting noise (right) is more amenable to the eye and much simpler to remove by a secondary filter

Fig. 7 Two-dimensional example for averages in voxels: The noisy input (**a**) is filtered in each vertex by path space filtering (**b**) yielding a splotchy result that is difficult to filter. Instead, the new method filters in each voxel, resulting in block artifacts (**c**). Additionally jittering before accumulation and lookup resolves the artifacts in noise that is simple to filter (**d**)

3.2 Accumulation and Lookup in a Hash Table

The averages in each voxel can be calculated in two different ways: First, each voxel can gather radiance of all included vertices. This process may run in parallel over all voxels and does not require any synchronization. On the other hand, a list of voxels as well as a list of vertices per voxel must be maintained. The second way to calculate the average radiance in a voxel runs in parallel over all vertices: Each vertex atomically adds its contribution $L_i(x', \omega) f_i(x', \omega) \cos\theta_{x'}$ to a running sum of the voxel and increments the counter of the voxel. Dividing the sum by the counter yields the average. While the latter approach requires atomic operations, it does not involve the maintenance of any lists. Furthermore, the summation can be parallelized over the paths or over the vertices, matching the parallelization scheme of typical light transport simulations. Finally, parallelization per path or per vertex exposes more parallelism, and therefore the second approach significantly outperforms the first one on modern *graphics processing units* (GPUs).

Accumulation with the latter approach needs a mapping from the key of a vertex to the voxel with its running sum and counter. Typically, the set of voxels is sparse since vertices are mostly on two-dimensional surfaces in three-dimensional space. Additional components of the key increase sparsity even further.

Hash tables provide such a mapping in constant time for typical sets of keys: First, a hash of the key is calculated using a fast hash function. A modulo operator then wraps this hash into the index range of the table cells. Since both the hash function as well as the modulo operator are not bijective, different keys may be mapped to the same index. Therefore, an additional check for equality of keys is required, and keys must also be stored in the table. Sect. 3.2.1 details a cheaper alternative for long keys.

Upon index collision with a different key, linear probing subsequently increments the index, checking if the table cell at the updated index is empty or occupied by an entry with same key. There exist various other collision resolution methods that improve upon several aspects of linear probing and have proven to be more efficient in certain use cases, especially for hash tables with high occupancy. On the other hand, we do not primarily aim to minimize the size of the hash table, and our experiments show that linear probing comes with a negligible overhead if the table is sufficiently

large. We restrict the number of steps taken for linear probing to avoid performance penalties of extreme outliers, and resort to the unfiltered contribution of the vertex if the number of steps exceeds this limit. So far, our choices for the table size and number of steps so extremely rarely resulted in such failures that further improvement has been deemed unnecessary. Sect. 3.2.2 broadens the application of linear probing from only collision resolution to an additional search for similar voxels.

3.2.1 Fingerprinting

Instead of storing and comparing the rather long keys, we calculate a shorter fingerprint [30, 34] from a second, different hash function of the same key and use it for this purpose, see Algorithm 1. Using a sentinel value that cannot be a fingerprint, we can furthermore mark empty cells.

Using fingerprints instead of the full keys is a tradeoff between correctness and performance: In theory, fingerprints of different keys may coincide. In practice, our choice of 32bit fingerprints never caused any collision in our evaluation of several test scenes and numerous simulations. Still, there is a certain probability of failure, and we deliberately favor the tiny probability of a failure over the performance penalty of storing and comparing full length keys.

Algorithm 1: Computation of the two hashes used for lookup. Note that the arguments of a hash function, which form the key, may be extended to refine clustering (denoted by "...", see Sect. 3.1.2).

Input: Location x of the vertex, the normal n, the position of the camera p_{cam}, and the scale s (see Sect. 3.1.1).
Output: Hash i to determine the position in the hash table and hash f for fingerprinting.

$l \leftarrow \texttt{level\_of\_detail}(|p_{cam} - x|)$
$x' \leftarrow x + \texttt{jitter}(n) \cdot s \cdot 2^l$
$l' \leftarrow \texttt{level\_of\_detail}(|p_{cam} - x'|)$
$\tilde{x} \leftarrow \left\lfloor \frac{x'}{s \cdot 2^{l'}} \right\rfloor$
$i \leftarrow \texttt{hash}(\tilde{x}, \ldots)$
$f \leftarrow \texttt{hash2}(\tilde{x}, n, \ldots)$

3.2.2 Searching by Linear Probing

As shown in Fig. 8, linear probing may be used to differentiate attributes of the light transport path at a finer resolution: For example, normal information may be included in the key handed to the fingerprinting hash function instead of already including it in the main key. This allows one to search for similar normals by linear probing.

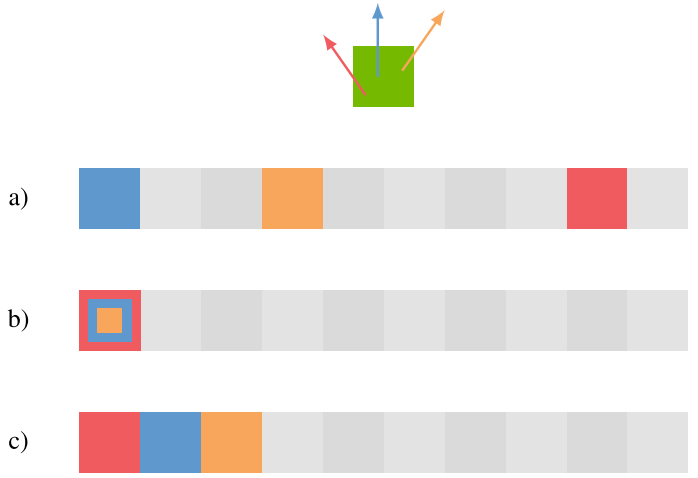


Fig. 8 Instead of including normals in the key (**a**) to differentiate contributions whose vertices fall into the same voxel (**b**), the fingerprint of a key may include normal information. This allows one to differentiate normal information by linear probing as shown in (**c**)

Note that due to completely unrelated voxels also possibly occupying neighboring cells in the hash table, searching with linear probing must go beyond those with mismatching fingerprints. Therefore, the method works best if both the number of additional contributions as well as the occupancy of the hash table are low.

3.3 Handling Voxels with a low Number of Vertices

Often, there exists a tiny number of voxels that contain very few vertices. Examples of such voxels include those that are only slightly overlapped by objects. Sects. 3.3.1 and 3.3.2 present two approaches that reduce variance in such voxels at an almost negligible cost.

3.3.1 Neighborhood Search

Accumulation in voxels by using quantized keys and a hash table requires one `atomicAdd` operation for each component of the radiance of each vertex as well as one `atomicAdd` or `atomicInc` operation for updating the counter of the voxel. The final average is computed with one additional non-atomic read operation per component for the sum and one for the counter. So, as long as access to the hash table happens in constant time, the calculation of the average also takes constant time. This is in sharp contrast to existing methods that compute sums or averages in

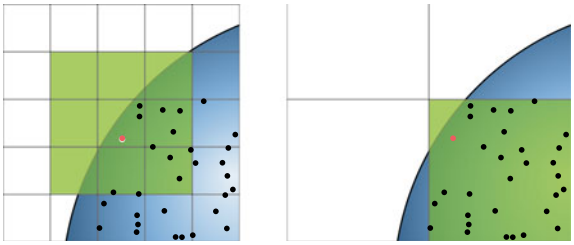


Fig. 9 Instead of searching the $\mathbf{n}^{\mathbf{d}}$ neighborhood in d dimensions (left), we utilize clustering resulting from quantization at a lower resolution to accumulate contributions, which allows for a single look up (right). We only resort to an additional neighborhood search in the rare case that the number of vertices in a voxel falls below a certain threshold

a spatial neighborhood which for each vertex takes linear time for a search within a given radius or (typically) logarithmic time for a fixed number of neighboring vertices.

Even if primarily only one average per voxel is computed and looked up, searching for neighboring voxels can still be valuable: In theory, increasing the resolution $s(k)$ and additionally searching for neighboring voxels may result in variance reduction similar to the one at a lower resolution, however then with a lower bias. Yet, the number of neighbors grows exponentially with the number of components of the key. Hence, such an approach is ruled out by the time constraints of real-time applications. Fig. 9 shows a comparison between neighborhood search and clustering by selecting a coarser resolution.

Neighborhood search is still very valuable as a fallback: If the number of vertices in a voxel falls below a certain threshold, we allow for an additional search. We observe that given an appropriate threshold, the number of such voxels is so low that the overhead is negligible while the perceptual improvement is clearly visible.

3.3.2 Multiresolution Accumulation

A special treatment of voxels with averages from only very few vertices is important for visual fidelity: Even if the number of voxels with a high variance is very low, they may be very visible, especially since their appearance is so different from the rest. Such voxels are very often found on the silhouette of objects. While one may not be able to identify them in still images, they become especially visible across frames. Besides searching in a local neighborhood to reduce variance in these cases (see Sect. 3.3.1), selecting a coarser resolution also effectively increases the number of vertices in the local average—at the price of an increased bias. Using more than one resolution at a time avoids the chicken-and-egg problem that arises from first selecting an appropriate resolution, and then, after accumulation according to this resolution, determining that it has been set too high or too low.

For simulations in interactive scenarios, one may also select the resolution based on information from previous frames, see Sect. 3.4.1.

3.4 Accumulation over Time

Reusing contributions over time dramatically increases efficiency. Attention should be paid to pitfalls and aspects of efficiency: Sect. 3.4.1 details the differences and similarities between filtering and integration across frames, Sect. 3.4.2 explains the handling of resolution changes across frames, and Sect. 3.4.3 is concerned with the amount of information stored over time to avoid running out of memory in the hash table.

3.4.1 Temporal Filtering and Temporal Integration

For static scenes, the averages will converge with an increasing number of frames. For dynamic environments, maintaining two sets of averaged contributions and combining them with an exponential moving average $c = \alpha \cdot c_{\text{old}} + (1 - \alpha) \cdot c_{\text{new}}$ is a common tradeoff between convergence and temporal adaptivity.

However, combining the averages c_{old} and c_{new} by an exponential moving average is not equivalent to temporal integration. Especially averages in voxels with relatively few samples do not converge. In fact, denoting N_{old} and N_{new} the number of vertices in the voxel in the previous and current frame, and setting $\alpha := \frac{N_{\text{old}}}{N_{\text{old}}+N_{\text{new}}}$ correctly integrates across frames. On the other hand, temporal integration is only possible if the underlying setting, including lighting conditions and object positions, remains unchanged across frames.

A first, simple heuristic is to accumulate samples over time up to a certain degree. This may be implemented using a fixed threshold for the number of samples and accumulating samples across frames until reaching it. Note that this heuristic is completely unaware of changes in the scene.

A second, more expensive heuristic builds upon temporal finite differences: A number of paths is re-evaluated with the same parameters, and the difference of their contribution to the original ones allows one to detect changes that affect the current voxel. Similar to the spatial finite differences in Sect. 3.1.2, the additional cost may be amortized across frames, and the number of samples used for finite differences as well as their influence on the balance between temporal adaptation and temporal integration must be carefully optimized. Note that averaging in voxels can be used for the samples used for finite differences, too. Figure 10 shows a comparison of temporal filtering, temporal integration and a hybrid that blends both based on temporal finite differences. A similar approach for screen space filtering has been explored in detail by Schied et al. [32].

(a) temporal filtering **(b)** temporal integration **(c)** hybrid

Fig. 10 In a scene with a moving light source, temporal filtering using an exponential moving average blurs shadow boundaries (**a**), and temporal integration averages out the entire shadow (**b**). Adaptively blending α between zero (for large temporal differences) and $\alpha := \frac{N_{\text{old}}}{N_{\text{old}}+N_{\text{new}}}$ (for no temporal differences) combines both and preserves sharp shadow boundaries (**c**). Note that jittering has deliberately been disabled here to enable simple distinction

3.4.2 Changing Resolution Across Frames

In many simulations, the camera is dynamic, and therefore the resolution of a voxel may change across frames if it depends on the position of the camera. Then accumulated contributions in a voxel at one resolution must be copied to a voxel at either a higher or lower resolution.

If the resolution in the new frame decreases, one can simply add up the contributions in voxels of higher resolution. Since we store sums and counters, both only need to be added individually.

If the resolution in the new frame increases, the contributions in voxels of lower resolution must be distributed to voxels at a higher resolution. Due to the lack of resolution, this case is much nuanced: On the one hand, using already collected contributions lowers variance, but on the other hand, the coarser resolution may become unpleasantly visible. One therefore needs to find a good compromise between the two, and set α in the exponential moving average accordingly.

Finite spatial or temporal differences can also be filtered across frames in a similar way.

3.4.3 Voxel Eviction Strategy

Evicting contributions of voxels which have not been queried for a certain period of time is necessary for larger scenes and changing camera. Besides the least recently used (LRU) eviction strategy, heuristics based on longer term observations are efficient.

A very simple method relies on replacing the most significant bits of the fingerprinting hash by a priority composed of for example the number of vertices in the voxel and last access time during temporal filtering. Thus the pseudo-randomly hashed least significant bits guarantee eviction to be uniformly distributed across the

scene, while the most significant bits ensure that contributions are evicted according to priority. This allows collision handling and eviction to be realized by a single `atomicMin` operation.

4 Results and Discussion

While filtering contributions at primary intersections with the proposed algorithm is quite fast, it only removes some artifacts of filtering in screen space. However, hashed path space filtering has been designed to target real-time light transport simulation: It is the only efficient option when screen space filtering fails or is not available, for example, when filtering after the first diffuse bounce (Fig. 11).

Filtering on non-diffuse surfaces requires to include additional parameters in the key and heuristics such as increasing the quantization in areas with non-diffuse materials to reduce the visible artifacts.

Filtering, and especially accumulating contributions, is always prone to light and shadow leaking (see Fig. 5), which is the price paid for performance. Some artifacts may be ameliorated by employing suitable heuristics as reviewed in [19, Sect. 2.1] and in Sect. 3.1.2.

The new algorithm filters incoherent intersections at HD resolution (1920×1080 pixels) in about 3ms on an NVIDIA Titan V GPU. Filtering primary intersections doubles the performance due to the more coherent memory access patterns.

The image quality is determined by the filter size, which balances noise versus blur as shown in Fig. 5. Both the number of collisions in the hash table and hence the performance of filtering depend on the size of the voxels, too. We found specifying the voxel size by s_0-times the projected size of a pixel most convenient. Since s_0 specifies the tradeoff between bias and variance reduction, its value highly depends on the scene and variance. Values between 4 and 16 may serve as a good starting point.

Note that maximum performance does not necessarily coincide with best image quality. The hash table size is chosen proportional to the number of pixels at target resolution such that potentially one vertex could be stored per pixel. In practice, filtering requires multiple vertices to coincide in a voxel, and therefore the occupancy of the hash table is rather low. Such a small occupancy improves the performance as it lowers the number of collisions and time spent for collision resolution.

While path space filtering dramatically reduces the noise at low sampling rates (see Fig. 1), some noise is added back by spatial jittering. Instead of selecting the first sufficiently diffuse vertex along a path from the camera, path space filtering can be applied at any vertex. For example, filtering at the second sufficiently diffuse vertex as shown in Fig. 11 resembles final gathering or local passes [17]. Furthermore, it is possible to filter in several vertices along the path at the same time. In fact, path space filtering trades variance reduction for controlled bias and is orthogonal to other filtering techniques. We therefore abstain from comparisons with these:

Fig. 11 Indirect illumination by hashed path space filtering only at the second bounce: At 16 paths per pixel (left), the variance of the integrand is dramatically reduced (right)

temporal anti-aliasing and complimentary noise filters in screen space are appropriate to further reduce noise [24]. A local smoothing filter [31] can even help reduce the error inherent in the approximation.

5 Conclusion

In combination with hardware accelerated ray tracing, our variance reduction technique enables visual fidelity of light transport simulation in real-time. Relying on only a few synchronizations during accumulation, path space filtering based on hashing scales on massively parallel hardware. Both accumulation as well as queries run in constant time per vertex. Neither the traversal nor the construction of a hierarchical spatial acceleration data structure is required. At the same time, the simplistic algorithm overcomes many restrictions of screen space filtering, does not require motion vectors, and enables variance reduction beyond the first intersection of a light transport path, including non-diffuse surfaces.

The hashing scheme still bears potential for improvement. For example, important hashes could be excluded from eviction by reducing the resolution that is accumulating their contributions at a coarser level. Other than selecting the resolution by the length of the path, path differentials and variance may be used to determine the appropriate resolution.

Besides the classic applications of path space filtering [19, Sect. 3] like multi-view rendering, spectral rendering, participating media, and decoupling anti-aliasing from shading, the adaptive hashing scheme can be applied to photon mapping [13, 17] and

irradiance probes in reinforcement learned importance sampling [6] in combination with final gathering. Since the first publication of this work as a technical report evolutions of the presented method have improved the efficiency of real-time ambient occlusion in massive scenes [9, 10], reinforcement learned importance sampling [29], and reservoir-based importance resampling [3] in light transport simulation.

Acknowledgements The authors thank Petrik Clarberg for profound discussions and comments.

References

1. Bako, S., Vogels, T., McWilliams, B., Meyer, M., Novák, J., Harvill, A., Sen, P., Derose, T., Rousselle, F.: Kernel-predicting convolutional networks for denoising Monte Carlo renderings. ACM Trans. Graph. **36**(4), 97:1–97:14 (2017). https://doi.org/10.1145/3072959.3073708
2. Bekaert, P., Sbert, M., Halton, J.: Accelerating path tracing by re-using paths. In: Debevec, P., Gibson, S. (eds.) Eurographics Workshop on Rendering. The Eurographics Association (2002). https://doi.org/10.2312/EGWR/EGWR02/125-134
3. Boissé, G.: World-space spatiotemporal reservoir reuse for ray-traced global illumination. ACM Trans. Graph. **40**(6) (2021)
4. Chaitanya, C., Kaplanyan, A., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., Aila, T.: Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. ACM Trans. Graph. **36**(4), 98:1–98:12 (2017). https://doi.org/10.1145/3072959.3073601
5. Cigolle, Z., Donow, S., Evangelakos, D., Mara, M., McGuire, M., Meyer, Q.: A survey of efficient representations for independent unit vectors. J. Comput. Graph. Tech. (JCGT) **3**(2), 1–30 (2014). http://jcgt.org/published/0003/02/01/
6. Dahm, K., Keller, A.: Learning light transport the reinforced way. In: Owen, A., Glynn, P. (eds.) Monte Carlo and Quasi-Monte Carlo Methods. MCQMC 2016. Proceedings in Mathematics & Statistics, vol. 241, pp. 181–195. Springer, Berlin (2018)
7. Dammertz, H.: Acceleration methods for ray tracing based global illumination. Ph.D. thesis, Universität Ulm (2011)
8. Dietrich, A., Slusallek, P.: Adaptive spatial sample caching. In: 2007 IEEE Symposium on Interactive Ray Tracing, pp. 141–147 (2007). https://doi.org/10.1109/RT.2007.4342602
9. Gautron, P.: Real-time ray-traced ambient occlusion of complex scenes using spatial hashing. In: Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks, SIGGRAPH '20. Association for Computing Machinery, New York, USA (2020)
10. Gautron, P.: Practical spatial hash map updates. In: Ray Tracing Gems II: Next Generation Real-Time Rendering with DXR. Vulkan, and OptiX, pp. 659–671. Apress, Berkeley, CA (2021)
11. Gautron, P., Droske, M., Wächter, C., Kettner, L., Keller, A., Binder, N., Dahm, K.: Path space similarity determined by Fourier histogram descriptors. In: ACM SIGGRAPH 2014 Talks, SIGGRAPH '14, pp. 39:1–39:1. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2614106.2614117
12. Georgiev, I., Křivánek, J., Davidovič, T., Slusallek, P.: Light transport simulation with vertex connection and merging. ACM Trans. Graph. **31**(6), 192:1–192:10 (2012)
13. Hachisuka, T., Jensen, H.: Stochastic progressive photon mapping. In: SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers, pp. 1–8. ACM (2009)
14. Hachisuka, T., Jensen, H.: Parallel progressive photon mapping on GPUs. SIGGRAPH Sketches (2010). https://doi.org/10.1145/1899950.1900004
15. Hachisuka, T., Pantaleoni, J., Jensen, H.W.: A path space extension for robust light transport simulation. ACM Trans. Graph. **31**(6) (2012). https://doi.org/10.1145/2366145.2366210

16. Havran, V., Herzog, R., Seidel, H.P.: Fast final gathering via reverse photon mapping. Comput. Graph. Forum **24**(3), 323–332 (2005)
17. Jensen, H.: Realistic Image Synthesis Using Photon Mapping. AK Peters (2001)
18. Keller, A.: Quasi-Monte Carlo Methods for Photorealistic Image Synthesis. Ph.D. thesis, University of Kaiserslautern, Germany (1998)
19. Keller, A., Dahm, K., Binder, N.: Path space filtering. In: Cools, R., Nuyens, D. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2014, pp. 423–436. Springer, Berlin (2016)
20. Kontkanen, J., Räsänen, J., Keller, A.: Irradiance filtering for Monte Carlo ray tracing. In: Talay, D., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2004, pp. 259–272. Springer, Berlin (2004)
21. Lafortune, E., Willems, Y.: Bi-directional path tracing. In: Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics' 93) (1998)
22. Ma, V., McCool, M.: Low latency photon mapping using block hashing. In: Ertl, T., Heidrich, W., Doggett, M. (eds.) SIGGRAPH/Eurographics Workshop on Graphics Hardware. The Eurographics Association (2002). https://doi.org/10.2312/EGGH/EGGH02/089-098
23. Mara, M., Luebke, D., McGuire, M.: Toward practical real-time photon mapping: efficient GPU density estimation. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D'13) (2013). https://casual-effects.com/research/Mara2013Photon/index.html
24. Mara, M., McGuire, M., Bitterli, B., Jarosz, W.: An efficient denoising algorithm for global illumination. In: ACM SIGGRAPH/Eurographics High Performance Graphics, p. 7 (2017). http://casual-effects.com/research/Mara2017Denoise/index.html
25. McCool, M.: Anisotropic diffusion for Monte Carlo noise reduction. ACM Trans. Graph. **18** (2002). https://doi.org/10.1145/318009.318015
26. Müller, R., McWilliams, B., Rousselle, F., Gross, M., Novák, J.: Neural importance sampling. ACM Trans. Graph. **38**(5), 145:1–145:19 (2019)
27. Müller, T., Gross, M., Novák, J.: Practical path guiding for efficient light-transport simulation. Comput. Graph. Forum **36**(4), 91–100 (2017)
28. Munkberg, J., Hasselgren, J., Clarberg, P., Andersson, M., Akenine-Möller, T.: Texture space caching and reconstruction for ray tracing. ACM Trans. Graph. **35**(6), 249:1–249:13 (2016). https://doi.org/10.1145/2980179.2982407
29. Pantaleoni, J.: Online path sampling control with progressive spatio-temporal filtering (2020)
30. Rabin, M.: Fingerprinting By Random Polynomials. Center for Research in Computing Technology, Harvard University, Technical report (1981)
31. Schied, C., Kaplanyan, A., Wyman, C., Patney, A., Chaitanya, C., Burgess, J., Liu, S., Dachsbacher, C., Lefohn, A., Salvi, M.: Spatiotemporal variance-guided filtering: Real-time reconstruction for path-traced global illumination. In: Proceedings of High Performance Graphics, HPG '17, pp. 2:1–2:12. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3105762.3105770
32. Schied, C., Peters, C., Dachsbacher, C.: Gradient estimation for real-time adaptive temporal filtering. Proc. ACM Comput. Graph. Interact. Tech. **1**(2) (2018)
33. Sen, P., Zwicker, M., Rousselle, F., Yoon, S.E., Kalantari, N.: Denoising your Monte Carlo renders: recent advances in image-space adaptive sampling and reconstruction. In: ACM SIGGRAPH 2015 Courses, SIGGRAPH '15, pp. 11:1–11:255. ACM, New York, NY, USA (2015). https://doi.org/10.1145/2776880.2792740
34. Slaney, M., Casey, M.: Locality-sensitive hashing for finding nearest neighbors [lecture notes]. IEEE Signal Process. Mag. **25**(2), 128–131 (2008). https://doi.org/10.1109/MSP.2007.914237
35. Veach, E.: Robust Monte Carlo Methods for Light Transport Simulation. Ph.D. thesis, Stanford University (1997)
36. West, R., Georgiev, I., Gruson, A., Hachisuka, T.: Continuous multiple importance sampling. ACM Trans. Graph. (Proceedings of SIGGRAPH) **39**(4) (2020)
37. Zwicker, M., Jarosz, W., Lehtinen, J., Moon, B., Ramamoorthi, R., Rousselle, F., Sen, P., Soler, C., Yoon, S.E.: Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. Comput. Graph. Forum **34**(2), 667–681 (2015)

A fresh Take on 'Barker Dynamics' for MCMC

Max Hird, Samuel Livingstone, and Giacomo Zanella

Abstract We study a recently introduced gradient-based Markov chain Monte Carlo method based on 'Barker dynamics'. We provide a full derivation of the method from first principles, placing it within a wider class of continuous-time Markov jump processes. We then evaluate the Barker approach numerically on a challenging ill-conditioned logistic regression example with imbalanced data, showing in particular that the algorithm is remarkably robust to irregularity (in this case a high degree of skew) in the target distribution.

Keywords MCMC · Markov chains · Barker dynamics · Langevin dynamics · Metropolis–Hastings

1 Introduction

For over half a century now Markov chain Monte Carlo has been used to sample from and compute expectations with respect to unnormalised probability distributions [16]. The idea is to construct a Markov chain for which a distribution of interest is invariant. Provided that the chain is π-irreducible and aperiodic (see e.g. [20]), then the distribution of X_n, the nth point in the chain, will approach the invariant distribution as $n \to \infty$, and ergodic averages from the chain can be used to approximate desired integrals.

M. Hird · S. Livingstone (✉)
Department of Statistical Science, University College London, Gower Street, London WC1E 6BT, UK
e-mail: samuel.livingstone@ucl.ac.uk

M. Hird
e-mail: max.hird.19@ucl.ac.uk

G. Zanella
Department of Decision Sciences, BIDSA and IGIER, Bocconi University, Via Roentgen 1, 20136 Milan, Italy
e-mail: giacomo.zanella@unibocconi.it

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_8

Restricting attention to $\mathsf{X} \subset \mathbb{R}^d$, one way to confirm that a distribution on X with density $\pi(x)$ is invariant for a Markov chain with transition kernel $Q(x, A) := \int_A q(x, y)dy$ is to establish that the equation

$$\frac{\pi(y)q(y, x)}{\pi(x)q(x, y)} := t(x, y) = 1 \tag{1}$$

holds for all $x, y \in \mathsf{X}$ such that $\pi(x)q(x, y) > 0$, and that $\pi(y)q(y, x) = 0$ elsewhere. These are the well-known *detailed balance* equations. The celebrated Metropolis–Hastings algorithm [10, 16] is built on the idea of coercing a Markov chain into having a specified invariant distribution. This is achieved through what will be called a *balancing function* in this article. Consider the scenario in which π is not invariant for Q, meaning (1) does not hold. A new kernel can be created which in fact *does* satisfy Eq. (1) by setting

$$p(x, y) := g(t(x, y))q(x, y), \tag{2}$$

where $g(t)$ satisfies

$$g(t) = tg(1/t) \tag{3}$$

whenever $t > 0$, and $g(0) := 0$. By noting that $\pi(x)q(x, y)t(x, y) = \pi(y)q(y, x)$ and $t(y, x) = 1/t(x, y)$, it is easily seen that

$$\begin{aligned} \pi(x)p(x, y) &= \pi(x)q(x, y)g(t(x, y)) \\ &= \pi(x)q(x, y)t(x, y)g(1/t(x, y))) \\ &= \pi(y)q(y, x)g(t(y, x)) \\ &= \pi(y)p(y, x) \end{aligned} \tag{4}$$

as required.

The problem, however, with taking the above strategy is that there is no guarantee that $\int p(x, y)dy = 1$, in fact this is extremely unlikely to be the case. More steps must be taken, therefore, to create a Markov process. The Metropolis–Hastings solution is to restrict to balancing functions that satisfy $g(t) \leq 1$ for all $t \in [0, \infty)$. This ensures that the kernel $K(x, A) := \int_A p(x, y)dy$ satisfies $K(x, \mathsf{X}) \leq 1$. The remaining probability mass can then be found by simply adding a *rejection step*, meaning that with probability $1 - K(x, \mathsf{X})$ the chain remains at its current point x.

There is, however, another way to create a Markov process from $p(x, y)$, without resorting to the Metropolis–Hastings approach. This consists of defining a continuous time Markov jump process in which jumps from the point x occur with intensity

$$\lambda(x) = \int p(x, y)dy, \tag{5}$$

and the jump location y is sampled from a distribution with density $p(x, y)/\lambda(x)$. The function $p(x, y)$ then describes the rate at which the process jumps from x to y, and (4) indicates that the process is π-invariant. The challenge associated with this second approach is that the integral (5) will often be intractable, meaning that simulating the process is not straightforward. Here we describe a solution that is outlined in the recent contribution [14], through a judicious choice of g and a suitable approximation to $t(x, y)$. It should of course be noted that in the case of finite X then (5) becomes a sum, and so the process can be exactly simulated. We do not consider this setting here, but direct the interested reader to [18], in which the approach is elegantly described, following earlier work in [23].

In the next section we discuss Barker's accept-reject rule, an early Markov chain Monte Carlo method from which we draw inspiration, before covering the general approach to the design of Markov jump processes with a prescribed invariant distribution in Sect. 3. It is here that we derive a Markov process that approximately preserves a given distribution, and show that the Barker balancing function is the only choice giving rise to such a process. In Sect. 4 we reveal the Barker proposal scheme, in which this new process is used as a proposal mechanism within a Metropolis–Hastings algorithm. In Sect. 5 we discuss the merits of using this new algorithm, by comparing it to suitable alternatives both theoretically and numerically, in the latter case using a challenging logistic regression example with imbalanced categorical data.

2 Barker's Rule and the Peskun Ordering

Readers who are familiar with the Metropolis–Hastings algorithm will naturally gravitate towards the choice of balancing function $g(t) = \min(1, t)$ in (2), resulting in the familiar Hastings acceptance probability

$$g_H(t(x, y)) = \min\left(1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right). \tag{6}$$

It should be noted, however, that several other choices of g are possible. One alternative proposed by Barker in [3] is $g(t) = t/(1 + t)$, resulting in the acceptance probability

$$g_B(t(x, y)) = \frac{\pi(y)q(y, x)}{\pi(x)q(x, y) + \pi(y)q(y, x)} \tag{7}$$

after multiplying the numerator and denominator by $\pi(x)q(x, y)$. In the case $q(x, y) = q(y, x)$ this further reduces to $\pi(y)/(\pi(x) + \pi(y))$. Note that both g_H and g_B satisfy $g \leq 1$.

The reason that g_H is preferred to g_B in the context of the Metropolis–Hastings algorithm is due to the work of Peskun [17] and Tierney [22], which established that for the same choice of $q(x, y)$ the acceptance rate g_H will result in Markov chains

that produce ergodic averages with smallest asymptotic variance. A key part of the argument is that g_H will maximize the probability of moving from x to y, for any $y \neq x$. When comparing (6) and (7) this is easy to see, as

$$g_B(t(x, y)) = \frac{\pi(y)q(y, x)}{\pi(x)q(x, y) + \pi(y)q(y, x)} \leq \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}$$

whenever $\pi(x)q(x, y) > 0$. Combining with the fact that $g_B \leq 1$ gives that $g_B \leq g_H$ for every value of $t(x, y)$.

It is important to emphasize, however, that the above discussion and conclusions about the optimality of g_H are confined to the scenario in which (2) is used to create a Metropolis–Hastings algorithm. It no longer applies to the setting in which the function $p(x, y)$ is used to define a Markov jump process with transition rates given by (5). Furthermore, in this case the stipulation that $g \leq 1$ is not required. This presents an opportunity to consider not just g_H but also alternatives such as g_B and others when designing jump processes of the type described in Sect. 1.

3 Designing Jump Processes Through a Balancing Function

The dynamics of a jump process for which transitions from x to y occur at the rate $p(x, y)$ are as follows: if the current point at time t is $X_t = x \in \mathsf{X}$, the process will remain at x for an exponentially distributed period of time $\tau \sim \text{Exp}(\lambda(x))$, with $\lambda(x)$ defined in (5), before moving to the next point using the Markov 'jump' kernel J, defined for any event A as

$$J(x, A) := \int_A \frac{p(x, y)}{\lambda(x)} dy. \tag{8}$$

In order to simulate such a process, we must therefore be able to compute $\lambda(x) := \int p(x, y)dy$, and also to simulate from $J(x, \cdot)$, for any $x \in \mathsf{X}$.

Note, however, that if $\lambda(x)$ were constant, we could simply use the jump kernel J directly and simulate a discrete-time Markov chain. To see this, note that in general the jump kernel will have invariant density proportional to $\lambda(x)\pi(x)$, as the equation

$$\lambda(x)\pi(x)\frac{p(x, y)}{\lambda(x)} = \lambda(y)\pi(y)\frac{p(y, x)}{\lambda(y)}$$

simplifies to $\pi(x)p(x, y) = \pi(y)p(y, x)$, which holds by design. If $\lambda(x) = \lambda$ then the above equation simply shows that J is π-reversible. We can therefore either simulate the continuous-time process with constant jump rate λ, or just ignore this step and take J as the kernel of a discrete-time Markov chain. In Sect. 3.1, we show that making a careful approximation to $t(x, y)$ allows just such a constant jump rate process to be found.

3.1 Tractability Through a 1st Order Approximation of $t(x, y)$

Recalling that $p(x, y) = g(t(x, y))q(x, y)$, the jump rate $\lambda(x)$ is the integral

$$\int g(t(x, y))q(x, y)dy,$$

which in general will not be tractable. A natural starting point for simplifying the problem is to restrict to the family of transition densities for which $q(x, y) = q(y, x)$, and further to the random walk case $q(x, y) = q(y - x)$. When this choice is made then $t(x, y) = \pi(y)/\pi(x)$. Restricting for now to $\mathsf{X} \subset \mathbb{R}$, a first order approximation of this ratio can be constructed using a Taylor series expansion as

$$\pi(y)/\pi(x) = \exp\{\log \pi(y) - \log \pi(x)\} \approx \exp\{(y - x)\nabla \log \pi(x)\}$$

for y suitably close to x. The purpose of using this approximation is that y now only enters the expression through the difference term $z := y - x$, and furthermore it holds that

$$t_x^*(z) := e^{z\nabla \log \pi(x)} = 1/e^{-z\nabla \log \pi(x)} = 1/t_x^*(-z). \tag{9}$$

Since $q(x, y) = q(z)$ we can therefore express the entire integral as

$$\lambda^*(x) := \int_{-\infty}^{\infty} g(t_x^*(z))q(z)dz.$$

By first writing $\lambda^*(x)$ as the sum of two integrals over the disjoint regions $(\infty, 0]$ and $[0, \infty)$, then switching the limits of integration through a change of variables in the first of these and finally re-combining, we arrive at the expression

$$\begin{aligned}
\lambda^*(x) &= \int_{-\infty}^{0} g(t_x^*(z))q(z)dz + \int_{0}^{\infty} g(t_x^*(z))q(z)dz \\
&= \int_{0}^{\infty} \left[g(t_x^*(-z))q(-z) + g(t_x^*(z))q(z)\right] dz \\
&= \int_{0}^{\infty} \left[g(t_x^*(-z)) + g(t_x^*(z))\right] q(z)dz,
\end{aligned}$$

where the last line follows from the fact that $q(z) = q(-z)$. Using (9) and then the balancing property (3) reveals that

$$g(t_x^*(-z)) = g(1/t_x^*(z)) = g(t_x^*(z))/t_x^*(z),$$

meaning that setting $t_x^*(z) := t^*$ the term in square brackets inside the integral can be written $(1 + 1/t^*)g(t^*)$. Note that if this expression were in fact equal to a constant, then $\lambda^*(x)$ would become tractable, and furthermore it would not depend on x. The Barker rule is the *unique* (up to constant multiple) choice of balancing function for which this property holds. To see this, note that for any $c \neq 0$

$$(1 + 1/t^*)g(t^*) = c \iff g(t^*) = \frac{c}{1 + 1/t^*}.$$

Setting $c = 1$ and multiplying by t^*/t^* reveals the choice $g_B(t^*) = t^*/(1 + t^*)$, and furthermore

$$\lambda^*(x) = \int_0^\infty q(z)dz = \frac{1}{2},$$

using the facts that $q(z) = q(-z)$ and $\int q(z)dz = 1$. In fact the choice of c is irrelevant here as it simply acts as a constant multiple to the jump rate and does not enter into the jump kernel expression. We refer to the resulting Markov process as *Barker dynamics*.

3.2 A Skew-Symmetric Markov Transition Kernel

The family of *skew-symmetric* distributions on $\mathbb{R}$ has densities of the form

$$2F(\beta z)\phi(z), \tag{10}$$

where ϕ is a symmetric probability density function, F is a cumulative distribution function such that $F(0) = 1/2$ and F' is a symmetric density, and $\beta \in \mathbb{R}$ [2]. Choosing $\beta > 0$ induces positive skew and vice versa (setting $\beta = 0$ means no skew is induced). In fact βz can be replaced with more general functions of z, but the above suffices for our needs.

The jump kernel (8) with symmetric choice of q, the approximation t^* in (9) and Barker balancing function g_B leads to the Markov kernel

$$J^*(x, A) := \int_A 2g_B(\exp\{(y - x)\nabla \log \pi(x)\})q(y - x)dy \tag{11}$$

for any event A. Writing $F_L(z) := 1/(1 + e^{-z})$, the cumulative distribution function of the logistic distribution, and noting that $g_B(e^z) = F_L(z)$, the associated transition density can be written

$$j^*(x, x + z) = 2F_L(\beta_x z)q(z)$$

where

$$\beta_x := \nabla \log \pi(x).$$

We see, therefore, that the resulting transition is *skew-symmetric*, with the level of skew at the current state x determined by $\nabla \log \pi(x)$. Because of this, a convenient algorithm for drawing samples from this transition kernel exists, and consists of the following:

1. Draw $\xi \sim q(\cdot)$
2. Set $b = 1$ with probability $F_L(\beta_x \xi)$, otherwise set $b = -1$
3. Set $z = b\xi$
4. Return $x + z$.

The resulting draw is from the kernel $J^*(x, \cdot)$. To see this, note that the probability density associated with any z is

$$j^*(x, x + z) = q(z)F_L(\beta_x z) + q(-z)(1 - F_L(-\beta_x z)),$$

which gives the density associated with either drawing z and setting $b = 1$ or drawing $-z$ and setting $b = -1$. After noting that $q(z) = q(-z)$ and $1 - F_L(-\beta_x z) = F_L(\beta_x z)$ by the symmetry of the logistic distribution, this simplifies to

$$j^*(x, x + z) = 2F_L(\beta_x z)q(z)$$

as required. Figure 1 illustrates the inner workings of such a transition. It is natural to consider whether other choices of skewing function derived from other balancing functions can be used to produce such a Markov transition. It is shown in Appendix F of [14], however, this is not possible, more precisely it is shown that g_B is the

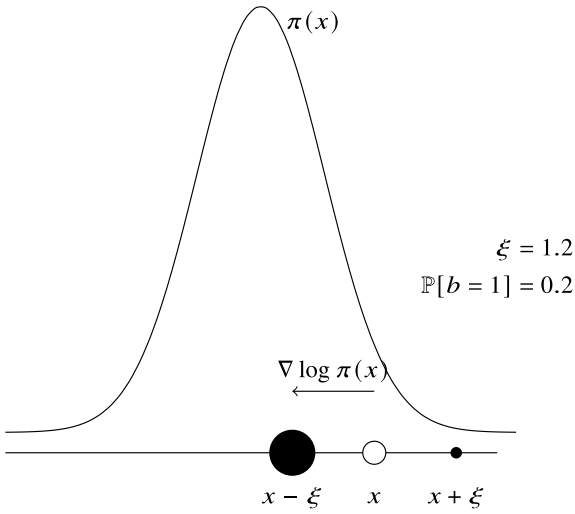


Fig. 1 A diagram of a typical draw from the transition kernel (11) using the algorithm outlined in Sect. 3.2. The white ball x is the current state, and the sizes of the black balls indicate the probability of moving to that point, given that the innovation drawn in step 1 is $\xi = 1.2$. A move in the direction of the gradient is clearly more probable

unique choice of balancing function leading to a skew-symmetric transition kernel when the first order approximation $t^*(x, y)$ is used in place of $t(x, y)$. This is in fact evident from the calculations of Sect. 3.1.

4 The Barker Proposal in $\mathsf{X} \subset \mathbb{R}^d$

The culmination of Sect. 3 is a Markov transition kernel (11) and an algorithm in Sect. 3.2 to draw samples from this kernel. Note, however, that this transition kernel will not in general have equilibrium distribution π, owing to the 1st order approximation used in Sect. 3.1. In some cases it might be reasonable to simply ignore this fact and use the method regardless, in the hope that any approximation error is small (the authors will discuss this approach in forthcoming work). The resolution we will adopt here, however, is to use the transition as a proposal within a Metropolis–Hastings algorithm. Note that the transition density can be written $j^*(x, y) \propto q(y - x)/(1 + e^{(x-y)\nabla \log \pi(x)})$ with $q(y - x) = q(x - y)$, meaning that the Metropolis–Hastings acceptance probability becomes

$$\alpha_1(x, y) = \min\left(1, \frac{\pi(y)(1 + e^{(x-y)\nabla \log \pi(x)})}{\pi(x)(1 + e^{(y-x)\nabla \log \pi(y)})}\right).$$

We have also restricted attention thus far to the one-dimensional setting, as extending to a d-dimensional transition kernel for $d > 1$ can be done in many different ways. It is natural to consider as a starting point a d-dimensional symmetric and centered density q. There are, however, many different ways to introduce the *skewing* mechanism into a d-dimensional distribution, which is done in one dimension through the variable $b \in \{-1, 1\}$. We consider two here, which we believe to be natural generalizations, and of which one is in fact clearly preferable to the other. The first is to simply introduce the same variable b, and after drawing $\xi \sim q(\cdot)$, set $\mathbb{P}[b = 1] = F_L(\beta_x^T \xi)$, where $\beta_x := \nabla \log \pi(x)$ as in Sect. 3.2. The only difference between this and the one-dimensional case is that now β_x and ξ are d-dimensional vectors, meaning the scalar product is replaced by an inner product. This procedure is a single global skewing of the initial symmetric distribution q.

It turns out, however, that a much more favorable approach is to skew each dimension individually. This involves defining $b \in \{-1, 1\}^d$, and setting $\mathbb{P}[b_i = 1] = F_L(\beta_{x,i}\xi_i)$ for $i \in \{1, \ldots, d\}$, where $\beta_{x,i} := \partial \log \pi(x)/\partial x_i$, the ith partial derivative of $\log \pi(x)$. This approach allows a much more flexible level of skewing to be applied to the base distribution q. In fact, once the initial $\xi \sim q(\cdot)$ is drawn, the first approach only considers two possible candidate moves: $x + \xi$ and $x - \xi$. In a high dimensional setting it may not be that either of these candidate moves is particularly favourable in terms of helping the chain mix. By contrast, the second approach allows for 2^d possible moves after ξ has been sampled. Figure 2 illustrates how this increased flexibility can result in much more favourable transitions.

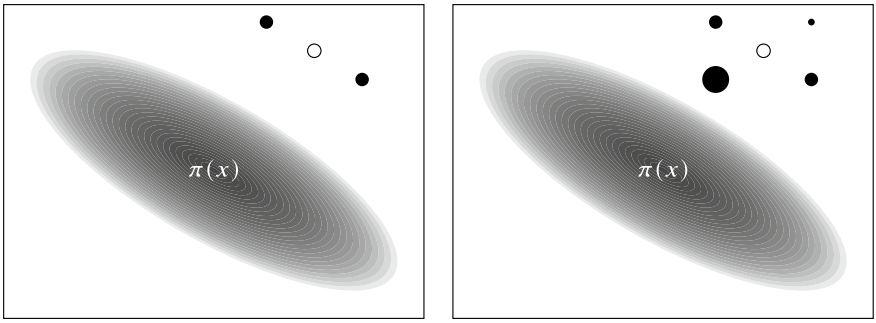


Fig. 2 A typical draw from the two different multi-dimensional transition kernels described in Sect. 4 when $d = 2$. The white ball x is the current state, and the sizes of the black balls indicate the probability of moving to each candidate point after the initial innovation ξ has been drawn. Using the first variant (left-hand side) only two moves are possible, neither of which move the chain closer to the high probability region of π. By contrast, using the second variant (right-hand side) 2^d moves are possible, and the most likely of these will move the chain in a favorable direction

One can make the comparison between the two approaches more concrete. In [14], it is shown that the asymptotic variance of the first d-dimensional version of the Barker proposal will be at least half as large as that of a random walk Metropolis algorithm. As such, using scaling arguments based on limiting diffusion approximations, it can be shown that $O(d)$ iterations of the algorithm are needed to achieve estimates of a fixed level of precision as $d \to \infty$. By contrast, in the same work it is shown that only $O(d^{1/3})$ iterations are needed for the second version to achieve the same goal. This is akin to the Metropolis-adjusted Langevin algorithm, another popular gradient-based Metropolis–Hastings algorithm (e.g. [19]). When referring to the Barker method in d-dimensions, from this point forward we will exclusively refer to the second approach described in this section. A single transition of the resulting d-dimensional Metropolis–Hastings algorithm with current state x is given below.

1. Draw $\xi \sim q(\cdot)$
2. For $i \in \{1, \ldots, d\}$ set $b_i = 1$ with probability $(1 + e^{-\beta_{x,i}\xi_i})^{-1}$, otherwise set $b_i = -1$, where $\beta_{x,i} := \partial \log \pi(x)/\partial x_i$
3. Set $y := x + b \cdot \xi$, where $a \cdot b = (a_1 b_1, a_2 b_2, \ldots, a_d b_d)$ defines the element-wise product of two vectors $a = (a_1, \ldots, a_d)$ and $b = (b_1, \ldots, b_d)$ in $\mathbb{R}^d$
4. Set the next state to be y with probability

$$\alpha_d(x, y) = \min\left(1, \frac{\pi(y)}{\pi(x)} \prod_{i=1}^{d} \frac{1 + e^{(x_i - y_i)\beta_{x,i}}}{1 + e^{(y_i - x_i)\beta_{y,i}}}\right),$$

otherwise remain at x.

We note that the algorithm requires the same ingredients as MALA, and has the same computational cost per iteration, which is dominated by the calculation of the gradient and target distribution. A simple function to run the Barker proposal in the R programming language is provided at https://github.com/gzanella/barker.

5 Why Use the Barker Algorithm?

Gradient-based MCMC methods are typically used because they perform well in high-dimensional settings. The Barker algorithm is no exception here, achieving the same $O(d^{-1/3})$ asymptotic efficiency as the popular Metropolis-adjusted Langevin algorithm (MALA) for suitably regular problems, where d represents the dimension of the state space [14]. The design of the Barker scheme, however, does differ from other gradient-based schemes such as MALA and Hamiltonian Monte Carlo (HMC). In both of the latter well-known approaches the gradient is incorporated through a deterministic drift, which depends linearly on $\nabla \log \pi(x)$. In MALA, for example, if the current point is x the proposal will be

$$y = x + \frac{h^2}{2}\nabla \log \pi(x) + h\xi,$$

where $\xi \sim N(0, 1)$ and $h > 0$. When the gradient is suitably regular and h well-chosen this transition can be very desirable; for example if π is Gaussian then the proposal becomes $y = (1 - h^2/2)x + h\xi$, leading to dynamics in which the chain drifts towards the centre of the space very quickly provided that $h^2 < 2$. In the same setting, however, it is immediately clear that choosing $h^2 > 2$ will lead to undesirable behaviour. The Barker proposal, by contrast, does not exhibit such a sharp cut-off between a good and bad choice of h in this example.

The above case is indicative of a much more general phenomenon that is well-known to practitioners, namely that popular gradient-based methods often produce fast-mixing Markov chains on a particular class of problems and provided that the tuning parameters of the algorithm are well-chosen, but that this class of problems is smaller than ideal, and that performance degrades rapidly when a poor choice of tuning parameters is made. This phenomenon is not only restricted to settings in which the MALA proposal becomes unstable (as in the Gaussian case), and means that it is also often difficult to tune the methods adaptively during the course of the simulation, an issue that is discussed in [14]. In that work the authors focus on characterizing *robustness to tuning*, providing a mathematical argument to show that for MALA and HMC performance is much more sensitive to the choice of proposal tuning parameters than for the Barker proposal.

5.1 Skewed Target Distributions

One scenario in which gradient-based algorithms can perform poorly is when the distribution of interest π exhibits considerable skew. To explore this phenomenon we first consider a simple one-dimensional model problem, before performing a more comprehensive numerical study on a challenging ill-conditioned logistic regression example. We will show that in both of these cases the Barker algorithm is considerably more robust to the level of skewness exhibited than other gradient-based schemes. In essence the challenge is that the gradient near the mode will diverge with the skewness of the distribution, causing pathologies in gradient-based proposals unless accounted for.

A model problem. Consider the family of skew-normal probability distributions π_η on $\mathbb{R}$ indexed by a skewness parameter $\eta > 0$. A given member of the family will have density $\pi_\eta(z) := 2\phi(z)\Phi(\eta z)$ where where ϕ and Φ are the density and cumulative distribution function of a standard normal distribution. Note that as η increases so does the skewness and that π_η becomes a truncated Gaussian truncated to be positive as $\eta \to \infty$. Take $x > 0$ larger than the mode of π_η, and set $y = 0$, noting that this implies $\text{sign}(\nabla \log \pi_\eta(x)) = -\text{sign}(\nabla \log \pi_\eta(y))$. The choice $y = 0$ is important only for the limiting result, in reality algorithmic difficulties will occur for any point in the neighborhood of zero for which the gradient is large and positive when $\eta \gg 0$. For these choices, as $\eta \to \infty$ it holds that $\pi_\eta(x) \to 2\phi(x)$, $\pi_\eta(y) \to 1/\sqrt{2\pi}$, and $\nabla \log \pi_\eta(x) \to -x$, whereas $\nabla \log \pi_\eta(y)) \to \infty$ as the density becomes increasingly skewed. Recall that the MALA proposal density is

$$\log q_\eta^M(z_1, z_2) := -\frac{1}{2h^2}\left(z_2 - z_1 - \frac{h^2}{2}\nabla \log \pi_\eta(z_1)\right)^2 - \frac{1}{2}\log\left(2\pi h^2\right).$$

This implies that $\log q_\eta^M(y, x) \to -\infty$ as $\eta \to \infty$, whereas $\log q_\eta^M(x, y)$ remains finite. As a consequence the reverse move from y to x becomes increasingly unlikely as η grows, causing the acceptance rate

$$\alpha_\eta^M(x, y) := \min\left(1, \frac{\pi_\eta(y) q_\eta^M(y, x)}{\pi_\eta(x) q_\eta^M(x, y)}\right)$$

to become arbitrarily small, such that $\alpha_\eta^M(x, y) \to 0$ as $\eta \to \infty$. The Barker proposal density is

$$\log q_\eta^B(z_1, z_2) := -\log\left(1 + \exp((z_1 - z_2)\nabla \log \pi_\eta(z_1)\right) + C$$

for some finite constant C. Since $(y - x)$ and $\nabla \log \pi_\eta(y)$ have opposite signs, their product tends to $-\infty$ in the same limit, meaning $\log q_\eta^B(y, x) \to C$. The acceptance rate $\alpha_\eta^B(x, y)$ for the Barker algorithm therefore remains stable and converges to a positive value in the same limit.

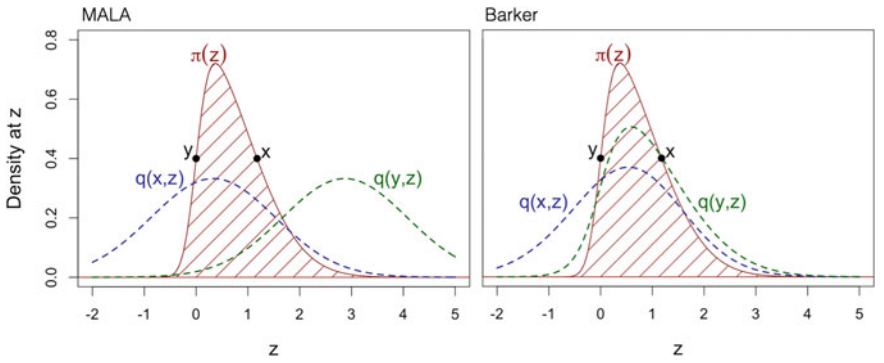


Fig. 3 The forward (blue) and reverse (green) proposal densities associated with two points separated by a mode for an example target distribution that contains skew. In the MALA case the current point x is quite unlikely under the reverse proposal density (green curve), whereas for the Barker algorithm this is not the case

Figure 3 provides some more intuition for the contrasting behaviour between the two methods in this example.

5.2 A Logistic Regression Example with Imbalanced Data

Skewed posterior distributions appear in many common modeling settings, but it is perhaps surprising that even seemingly simple logistic regression models can exhibit such a degree of skew that they pose a significant challenge to MCMC methods. This is despite the fact that the posterior distribution is strongly log-concave and the gradient is Lipschitz, meaning that several favorable results on the mixing properties of classical gradient-based algorithms can been established (e.g. [6–8]).

We consider an example using the arrythmia dataset from the UCI machine learning repository, available at https://archive.ics.uci.edu/ml/datasets/arrhythmia. The dataset consists of 452 observations of 279 different covariates. The modeling task is to detect the presence or absence of cardiac arrythmia. The data presents a challenge as there are many imbalanced categorical covariates with only a few observations in certain categories.

The number of predictors compared to the size of the dataset makes the problem highly ill-conditioned. To combat this we selected 25 imbalanced covariates and 25 others, meaning 50 covariates in total for our problem. The 25 imbalanced predictors were chosen from among the categorical covariates for which one category appeared two or fewer times in the dataset, whereas the remaining 25 were chosen from the remaining set. Despite this pre-processing the problem is still highly ill-conditioned and the maximum likelihood estimator is undefined, making a Bayesian approach very natural for the problem. We also note that despite the reduced number of covari-

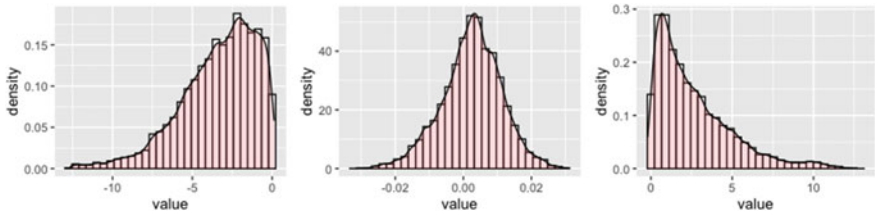


Fig. 4 Example marginal distributions for selected covariates using the output of the Barker algorithm, illustrating varying degrees of skew in different dimensions. The plots are shown for the raw data

ates the final problem is still of large enough dimension that simpler fitting methods will be ineffective, in line with the recommendations of [5]. The choice of dataset was inspired by [11], in which the authors highlight that imbalanced categorical data can cause problems for Markov chain Monte Carlo methods. In this case the result is a logistic regression posterior distribution with a pronounced level of skewness in certain dimensions, as shown in Fig. 4.

For the Barker scheme we choose a Gaussian $q(\cdot)$ (although we note anecdotally that ongoing work suggests that other choices may be preferable). With the goal of minimizing the degree of hand-tuning needed for each algorithm, we used an adaptive approach to choosing algorithmic tuning parameters, precisely Algorithm 4 of [1], which consists of a Robbins–Monro scheme for learning a single global scale λ and a covariance matrix Σ, which combine to form the pre-conditioning matrix $\lambda^2\Sigma$. We set the Robbins–Monro learning rate at time t to be $t^{-0.6}$. The matrix Σ can be dense or restricted to diagonal; the former allows correlations to be better navigated by the sampler, but the diagonal approach means less parameters must be learned during the simulation. A weakly-informative independent Gaussian prior with zero mean and variance 25 was chosen for each model parameter. It is also sometimes recommended in logistic regression problems to first *standardise* the covariates, transforming each to have zero mean and unit variance. This can have the effect of making the posterior more regular and as a consequence the inference less challenging, but is not always done by practitioners. In our case the scales by which the covariates were standardized range from ~0.05 to ~32.

The above considerations led us to four different testing scenarios for each algorithm: dense Σ with raw data, dense Σ with standardized data, diagonal Σ with raw data and diagonal Σ with standardized data. For each of these scenarios we compared the Barker proposal scheme with MALA, a classical gradient-based alternative, as a simple illustration of the different patterns of behavior that the two algorithms can exhibit.

Trace plots showing the performance of the MALA and Barker algorithms in each scenario are shown in Fig. 5. It is immediately clear that MALA struggles to reach equilibrium in 3 out of 4 scenarios, only really performing reasonably when

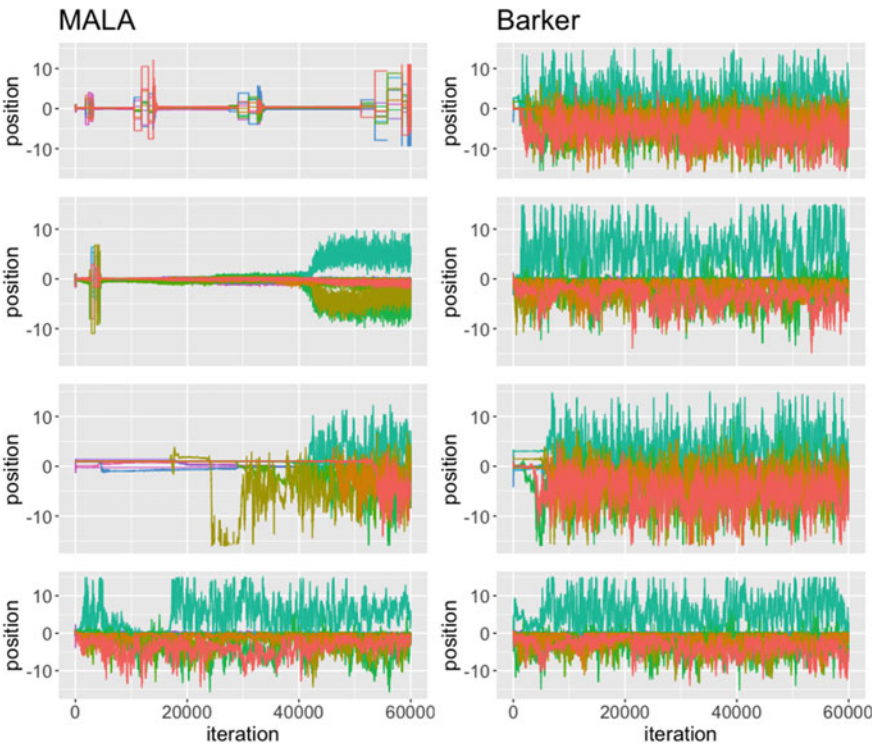


Fig. 5 A selection of trace plots from the MALA and Barker algorithms for the logistic regression example. 1st row: raw data with dense Σ; 2nd row: standardized data with dense Σ; 3rd row: raw data with diagonal Σ; 4th row: standardized data with diagonal Σ

Σ is diagonal and the data is standardized. As expected, standardizing the data aids performance, but it is perhaps surprising that the sampler also struggles in the dense Σ setting. By comparison, visually the Barker algorithm behaves reasonably in all scenarios. To evaluate the samplers at equilibrium and once the adaptation has stabilized, we examine effective sample sizes for each scenario in which equilibrium is visually reached after 30,000 iterations in Table 1. The effective sample sizes allow us to see that performance once equilibrium is reached is largely comparable between the two schemes once the scenario is favourable. The key strengths of the Barker approach in this example are its robustness to lack of standardization and robustness to different adaptation strategies.

Table 1 Minimum and median effective sample sizes (ESS) for the logistic regression example

Dataset	Algorithm	ESS (min., med.)
Raw	Barker (dense)	38.82, 156.67
Raw	Barker (diag.)	65.55, 164.67
Raw	MALA (dense)	N/a
Raw	MALA (diag.)	N/a
Standardised	Barker (dense)	53.36, 98.44
Standardised	Barker (diag.)	44.19, 101.51
Standardised	MALA (dense)	N/a
Standardised	MALA (diag.)	37.21, 87.14

6 Discussion

We have given a pedagogical treatment of the Barker proposal scheme, a new gradient-based MCMC algorithm that we argue has some desirable features when compared to classical gradient-based alternatives, namely its robustness (in a very general sense). There are numerous ways in which classical schemes such as MALA and HMC can be made more robust in different settings (e.g. [4, 12, 15, 21]), but these often introduce additional tuning parameters and can suffer from other issues, meaning that the quality of performance becomes very problem-specific. Another alternative approach are second-order methods that incorporate the Hessian of $\log \pi(x)$ in some way (e.g. [9, 13]), but generally the cost of their implementation is large, and can grow cubically with dimension. Based on the simplicity, scaling properties and robustness of the Barker proposal we argue that there are likely to be many realistic scenarios in which it proves useful, and in addition there is much room for the development of further algorithms within the general framework discussed in Sect. 3.

References

1. Andrieu, C., Thoms, J.: A tutorial on adaptive MCMC. Stat. Comput. **18**(4), 343–373 (2008)
2. Azzalini, A., Regoli, G.: Some properties of skew-symmetric distributions. Ann. Inst. Stat. Math. **64**(4), 857–879 (2012)
3. Barker, A.A.: Monte Carlo calculations of the radial distribution functions for a proton-electron plasma. Australian J. Phys. **18**(2), 119–134 (1965)
4. Brosse, N., Durmus, A., Moulines, É., Sabanis, S.: The tamed unadjusted Langevin algorithm. Stoch. Process. Their Appl. **129**(10), 3638–3663 (2019)
5. Chopin, N., Ridgway, J., et al.: Leave Pima Indians alone: binary regression as a benchmark for Bayesian computation. Stat. Sci. **32**(1), 64–87 (2017)
6. Dalalyan, A.S., Karagulyan, A.: User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient. Stoch. Process. Their Appl. **129**(12), 5278–5311 (2019)

7. Durmus, A., Moulines, E., et al.: Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. Ann. Appl. Prob. **27**(3), 1551–1587 (2017)
8. Dwivedi, R., Chen, Y., Wainwright, M.J., Yu, B.: Log-concave sampling: Metropolis-Hastings algorithms are fast! In: Conference on Learning Theory, pp. 793–797. PMLR (2018)
9. Girolami, M., Calderhead, B.: Riemann manifold Langevin and Hamiltonian Monte Carlo methods. J. R. Stat. Soc.: Ser. B (Statistical Methodology) **73**(2), 123–214 (2011)
10. Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika 97–109 (1970)
11. Johndrow, J.E., Smith, A., Pillai, N., Dunson, D.B.: MCMC for imbalanced categorical data. J. Am. Stat. Assoc. **114**(527), 1394–1403 (2019)
12. Livingstone, S., Faulkner, M.F., Roberts, G.O.: Kinetic energy choice in Hamiltonian/hybrid Monte Carlo. Biometrika **106**(2), 303–319 (2019)
13. Livingstone, S., Girolami, M.: Information-geometric Markov chain Monte Carlo methods using diffusions. Entropy **16**(6), 3074–3102 (2014)
14. Livingstone, S., Zanella, G.: The Barker proposal: combining robustness and efficiency in gradient-based MCMC (2019). arXiv:1908.11812
15. Lu, X., Perrone, V., Hasenclever, L., Teh, Y.W., Vollmer, S.: Relativistic Monte Carlo. In: Artificial Intelligence and Statistics, pp. 1236–1245. PMLR (2017)
16. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
17. Peskun, P.H.: Optimum Monte-Carlo sampling using Markov chains. Biometrika **60**(3), 607–612 (1973)
18. Power, S., Goldman, J.V.: Accelerated Sampling on Discrete Spaces with Non-Reversible Markov Processes (2019). arXiv:1912.04681
19. Roberts, G.O., Rosenthal, J.S.: Optimal scaling of discrete approximations to Langevin diffusions. J. R. Stat. Soc.: Ser. B (Statistical Methodology) **60**(1), 255–268 (1998)
20. Roberts, G.O., Rosenthal, J.S.: General state space Markov chains and MCMC algorithms. Probab. Surv. **1**, 20–71 (2004)
21. Roberts, G.O., Tweedie, R.L., et al.: Exponential convergence of Langevin distributions and their discrete approximations. Bernoulli **2**(4), 341–363 (1996)
22. Tierney, L.: A note on Metropolis-Hastings kernels for general state spaces. Ann. Appl. Prob. 1–9 (1998)
23. Zanella, G.: Informed proposals for local MCMC in discrete spaces. J. Am. Stat. Assoc. **115**(530), 852–865 (2020)

On the Selection of Random Field Evaluation Points in the p-MLQMC Method

Philippe Blondeel, Pieterjan Robbe, Stijn François, Geert Lombaert, and Stefan Vandewalle

Abstract Engineering problems are often characterized by significant uncertainty in their material parameters. A typical example coming from geotechnical engineering is the slope stability problem where the soil's cohesion is modeled as a random field. An efficient manner to account for this uncertainty is the novel sampling method called p-refined Multilevel Quasi-Monte Carlo (p-MLQMC). The p-MLQMC method uses a hierarchy of p-refined finite element meshes combined with a deterministic Quasi-Monte Carlo sampling rule. This combination yields a significant computational cost reduction with respect to classic Multilevel Monte Carlo. However, in previous work, not enough consideration was given to how to incorporate the uncertainty, modeled as a random field, in the finite element model with the p-MLQMC method. In the present work we investigate how this can be adequately achieved by means of the integration point method. We therefore investigate how the evaluation points of the random field are to be selected in order to obtain a variance reduction over the levels. We consider three different approaches. These approaches will be benchmarked on a slope stability problem in terms of computational runtime. We find that for a given tolerance the *local nested approach* yields a speedup up to a factor five with respect to the *non-nested approach*.

Keywords Multilevel Quasi-Monte Carlo · p-refinement · Karhunen-Loève expansion

P. Blondeel (✉) · P. Robbe · S. Vandewalle
Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium
e-mail: philippe.blondeel@kuleuven.be

P. Robbe
e-mail: pieterjan.robbe@kuleuven.be

S. Vandewalle
e-mail: stefan.vandewalle@kuleuven.be

S. François · G. Lombaert
Department of Civil Engineering, KU Leuven, Kasteelpark Arenberg 40, 3001 Leuven, Belgium
e-mail: stijn.francois@kuleuven.be

G. Lombaert
e-mail: geert.lombaert@kuleuven.be

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_9

1 Introduction

Starting from the work by Giles, see [7–9], we developed a novel multilevel method called p-refined Multilevel Quasi Monte Carlo (p-MLQMC), see [1]. Similar to classic Multilevel Monte Carlo, see [7], p-MLQMC uses a hierarchy of increasing resolution finite element meshes to achieve a computational speedup. Most of the samples are taken on coarse and computationally cheap meshes, while a decreasing number of samples are taken on finer and computationally expensive meshes. The major difference between classic Multilevel Monte Carlo and p-MLQMC resides in the refinement scheme used for constructing the mesh hierarchy. In classic Multilevel Monte Carlo (h-MLMC), an h-refinement scheme is used to build the mesh hierarchy, see, for example [4]. The accuracy of the model is increased by increasing the number of elements in the finite element mesh. In p-MLQMC, a p-refinement scheme is used to construct the mesh hierarchy. The accuracy of the model is increased by increasing the polynomial order of the element's shape functions while retaining the same number of elements. This approach reduces the computational cost with respect to h-MLMC, as shown in [1]. Furthermore, instead of using a random sampling rule, i.e., the Monte Carlo method, p-MLQMC uses a deterministic Quasi-Monte Carlo (QMC) sampling rule, yielding a further computational gain.

However, the p-MLQMC method presents the practitioner with a challenge. This challenge consists of adequately incorporating the uncertainty, modeled as a random field, into the finite element model. For classic Multilevel (Quasi)-Monte Carlo (h-ML(Q)MC) this is typically achieved by means of the midpoint method [11]. The model uncertainty is represented as scalars resulting from the evaluation of the random field at centroids of the elements. These scalars are then assigned to the elements. With this method, the uncertainty is modeled as being constant inside each element. In h-refined multilevel methods, the midpoint method intrinsically links the spatial resolution of the mesh with the spatial resolution of the random field. An h-refinement of the mesh will result in a finer representation of the random field. However, for the p-MLQMC method, the midpoint method cannot be used. This is because the refinement scheme used in p-MLQMC does not increase the number of elements. The p-MLQMC method makes use of the integration point method, see [13]. Scalars resulting from the evaluation of the random field at certain spatial locations are taken into account during numerical integration of the element stiffness matrices. With this method, the uncertainty varies inside each element. In the present work, we investigate how to adequately select the spatial locations used for the evaluation of the random field. Specifically, we distinguish three different approaches to how to select these random field evaluation points. The *Non-Nested Approach* (NNA), the *Global Nested Approach* (GNA), and the *Local Nested Approach* (LNA). We investigate how these approaches affect the variance reduction in the p-MLQMC method, and how the total computational runtime increases over the levels. These approaches will be benchmarked on a model problem which consists of a slope stability problem, which assesses the stability of natural or man made slopes. The uncertainty is located in the soil's cohesion, and is represented as a two-dimensional lognormal random field.

The paper is structured as follows. First we give a theoretical background motivating our research, and give a concise overview of the building blocks of p-MLQMC. Second, we present the three approaches. Hereafter, we shortly discuss the underlying finite element solver, and introduce the model problem. Last, we present the results obtained with p-MLQMC for the three different approaches. Here we focus on the variance reduction over the levels and the effect on the total computational runtime.

2 Theoretical Background

Multilevel Monte Carlo methods rely on a hierarchy of meshes in order to achieve a speedup with respect to Monte Carlo. This speedup is achieved by writing the expected value of a quantity of interest on a fine mesh as the expected value of a quantity of interest on a coarse mesh together with a series of correction terms that express the difference in expected value of the quantity of interest on two successive finer meshes. In particular, given the hierarchy of approximations $P_0, P_1, \ldots, P_{\mathrm{L}}$ for the quantity of interest P computed on an increasingly finer mesh, we have the telescopic sum identity,

$$\mathbb{E}[P_{\mathrm{L}}] = \mathbb{E}[P_0] + \sum_{\ell=1}^{\mathrm{L}} \mathbb{E}[P_\ell - P_{\ell-1}]. \tag{1}$$

This hierarchy of meshes can be obtained by applying an h-refinement scheme or a p-refinement scheme to a coarse mesh model. We opt for a hierarchy based on p-refinement. The hierarchy applied to a discretized model of the slope stability problem is shown in Fig. 1. Here, the finite element nodal points are represented as red dots. A more thorough discussion of the slope stability problem and the underlying finite element model is given in Sect. 4.

In the Multilevel Monte Carlo setting, the meshes in the hierarchy are commonly referred to as 'levels'. The coarsest mesh is referred to as level 0. Subsequent finer meshes are assigned the next cardinal number, e.g., level 1, level 2, ...

The number of samples to be taken on levels greater than 0 ($\ell > 0$), is proportional to the sample variance of the difference, $\mathbb{V}[\Delta P_\ell]$ with $\Delta P_\ell = P_\ell - P_{\ell-1}$ and P a chosen quantity of interest (QoI). It is only for determining the number of samples on level 0 that the sample variance $\mathbb{V}[P_\ell]$ is used. In order to obtain a decreasing number of samples per increasing level, i.e., $N_0 > N_1 > \cdots > N_{\mathrm{L}}$, it is necessary to have a variance reduction over the levels, i.e., $\mathbb{V}[\Delta P_1] > \mathbb{V}[\Delta P_2] > \cdots > \mathbb{V}[\Delta P_{\mathrm{L}}]$, and an increasing cost 'of one solve' per increasing level, i.e., $\mathcal{C}_0 < \mathcal{C}_1 < \cdots < \mathcal{C}_{\mathrm{L}}$. This variance reduction is only obtained when a strong positive correlation is achieved between the results of two successive levels. We have that

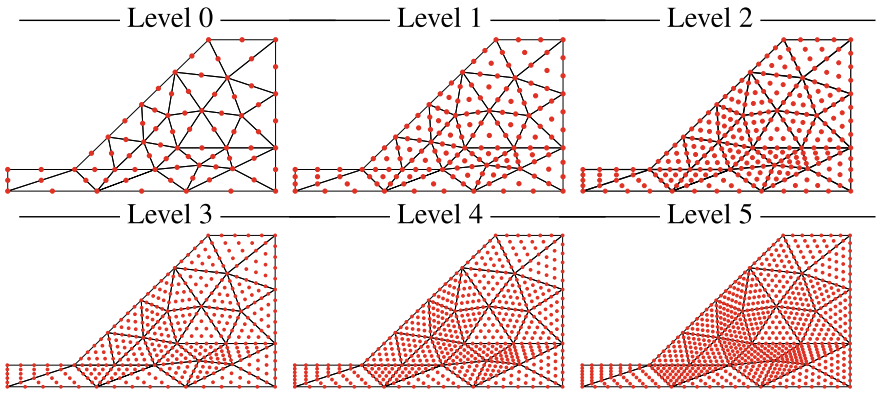


Fig. 1 p-refined hierarchy of approximations used for the slope stability problem

$$\begin{aligned}\mathbb{V}\left[\Delta P_\ell\right] &= \mathbb{V}\left[P_\ell - P_{\ell-1}\right] \\ &= \mathbb{V}\left[P_\ell\right] + \mathbb{V}\left[P_{\ell-1}\right] - 2\mathrm{cov}\left(P_\ell, P_{\ell-1}\right),\end{aligned} \tag{2}$$

where $\mathrm{cov}\left(P_\ell, P_{\ell-1}\right) = \rho_{\ell,\ell-1}\sqrt{\mathbb{V}\left[P_\ell\right]\mathbb{V}\left[P_{\ell-1}\right]}$ is the covariance between P_ℓ and $P_{\ell-1}$ with $\rho_{\ell,\ell-1}$ the correlation coefficient. The value of $\mathrm{cov}\left(P_\ell, P_{\ell-1}\right)$ must be larger than 0 to have a large variance reduction, and hence an efficient multilevel method.

In our p-MLQMC algorithm applied to a slope stability problem, see [1], the model uncertainty representing the soil's cohesion is located in the elastic constitutive matrix $\mathbf{D}$. It is taken into account at the locations of the quadrature points when computing the integral in the element stiffness matrices $\mathbf{K^e}$, i.e.,

$$\mathbf{K^e} = \int_{\Omega_e} \mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}\mathrm{d}\Omega_e. \tag{3}$$

This is calculated in practice as

$$\mathbf{K^e} = \sum_{i=1}^{|\mathbf{q}|} \mathbf{B}_i^{\mathrm{T}}\mathbf{D}_i\mathbf{B}_i\mathrm{w}_i, \tag{4}$$

where the matrix $\mathbf{B}_i = \mathbf{B}(\mathbf{q}^i)$ contains the derivatives of the shape functions, evaluated at the quadrature points $\mathbf{q}^i$, the matrix $\mathbf{D}_i = \mathbf{D}\left(\mathbf{x}^i, \omega\right)$ contains the model uncertainty computed at point $\mathbf{x}^i$, and w_i are the quadrature weights. The set of quadrature points $\mathbf{q}$ is expressed in a local coordinate system of the triangular reference element. The uncertainty in the matrix $\mathbf{D}_i$ is represented by a scalar originating from the evaluation of the random field at a carefully chosen spatial location. This approach is commonly referred to as the *integration point method* [13]. Note that here

the uncertainty is not constant in an element, i.e., $\mathbf{D}_1 \neq \mathbf{D}_2 \neq \cdots \neq \mathbf{D}_i$ in Eq. (4). The scalar used in the matrix $\mathbf{D}_i$ originates from the evaluation of the random field at spatial location $\mathbf{x}^i \in \mathbf{x}$, in a global coordinate system of the mesh, by means of the Karhunen-Loève (KL) expansion with stochastic dimension s, i.e.,

$$Z(\mathbf{x}, \omega) = \overline{Z}(\mathbf{x}) + \sum_{n=1}^{s} \sqrt{\theta_n}\xi_n(\omega)b_n(\mathbf{x}) \,, \tag{5}$$

where $\overline{Z}(\mathbf{x})$ is the mean of the field and $\xi_n(\omega)$ denote i.i.d. standard normal random variables. The symbols θ_n and $b_n(\mathbf{x})$ denote the eigenvalues and eigenfunctions respectively, which are the solutions of the eigenvalue problem $\int_D C(\mathbf{x}, \mathbf{y})b_n(\mathbf{y})\mathrm{d}\mathbf{y} = \theta_n b_n(\mathbf{x})$ with a given covariance kernel $C(\mathbf{x}, \mathbf{y})$. Note that in order to represent the uncertainty of the soil's cohesion in the considered slope stability problem, we do not use $Z(\mathbf{x}, \omega)$ but $\exp(Z(\mathbf{x}, \omega))$, see Sect. 4.

Our goal is to select evaluation points for Eq. (5), grouped in sets $\{\mathbf{x}_\ell\}_{\ell=0}^{\mathrm{L}}$, in order to ensure a good correlation between P_ℓ and $P_{\ell-1}$, i.e., such that the covariance, cov $(P_\ell, P_{\ell-1})$, is as large as possible, see Eq. (2). We distinguish three different approaches for selecting the evaluation points on the different levels, the *Non-Nested Approach* (NNA), the *Global Nested Approach* (GNA) and the *Local Nested Approach* (LNA). All the approaches start from the given sets of quadrature points on the different levels $\{\mathbf{q}_\ell\}_{\ell=0}^{\mathrm{L}}$. Note that the number of quadrature points per level increases, $|\mathbf{q}_0| < |\mathbf{q}_1| \cdots < |\mathbf{q}_{\mathrm{L}}|$. Given the sets $\{\mathbf{q}_\ell\}_{\ell=0}^{\mathrm{L}}$, we select evaluation points for the random field in a local coordinate system and group them in sets $\left\{\mathbf{x}_\ell^{\mathrm{local}}\right\}_{\ell=0}^{\mathrm{L}}$, with the condition that $|\mathbf{x}_\ell^{\mathrm{local}}| = |\mathbf{q}_\ell|$. The points in the sets $\left\{\mathbf{x}_\ell^{\mathrm{local}}\right\}_{\ell=0}^{\mathrm{L}}$ are then transformed to points in global coordinates, resulting in sets $\{\mathbf{x}_\ell\}_{\ell=0}^{\mathrm{L}}$. The points belonging to $\{\mathbf{x}_\ell\}_{\ell=0}^{\mathrm{L}}$ are then used in Eq. (1). Note that with the integration point method, the spatial resolution of the field is proportional to the number of quadrature points. Increasing the number of quadrature points will result in a finer resolution of the random field.

Before elaborating further upon these approaches, we first introduce the estimator used in our p-MLQMC algorithm. The estimator is given by

$$\begin{aligned} Q_{\mathrm{L}}^{\mathrm{MLQMC}} := & \frac{1}{R_0}\sum_{r=1}^{R_0}\frac{1}{N_0}\sum_{n=1}^{N_0} P_0(\mathbf{u}_0^{(r,n)}) + \\ & \sum_{\ell=1}^{\mathrm{L}}\frac{1}{R_\ell}\sum_{r=1}^{R_\ell}\left\{\frac{1}{N_\ell}\sum_{n=1}^{N_\ell}\left(P_\ell(\mathbf{u}_\ell^{(r,n)}) - P_{\ell-1}(\mathbf{u}_\ell^{(r,n)})\right)\right\}. \end{aligned} \tag{6}$$

It expresses the expected value of the quantity of interest on the finest level L as the sample average of the quantity of interest on the coarsest level, plus a series of correction terms. A particularity of MLQMC consists of the use of deterministic sample points per level $\mathbf{u}_\ell^{(r,n)}$ in the unit cube, i.e., $[0, 1]^s$, combined with an average over a number of shifts R_ℓ on each level ℓ. Averaging over the number of shifts is

performed in order to obtain unbiased estimates of the computed stochastic quantities. The representation of these uniform distributed quasi-Monte Carlo sample points in $[0, 1]^s$ is given by

$$\mathbf{u}^{(r,n)} = \text{frac}\,(\phi_2(n)\mathbf{z} + \Xi_r) \quad \text{for} \quad n \in \mathbb{N}, \tag{7}$$

where $\text{frac}\,(x) = x - (x), x > 0$, ϕ_2 is the radical inverse function in base 2, $\mathbf{z}$ is an s-dimensional vector of positive integers, $\Xi_r \in [0, 1]^s$ is the random shift with $r = 1, 2, \ldots, R_\ell$, and s the stochastic dimension. The representation of the points from Eq. (7) is known as a *shifted rank-1 lattice rule*. The generating vector $\mathbf{z}$ was constructed with the component-by-component (CBC) algorithm with decreasing weights, $\gamma_j = 1/j^2$, see [10]. In the scope of this work, the uniform quasi-Monte Carlo sample points $\mathbf{u}_\ell^{(r,n)}$ are mapped from $[0, 1]^s$ to $\mathbb{R}^s$ by means of the inverse of the univariate standard normal cumulative distribution function, $\Phi^{-1}(\cdot)$. The standard normal distributed quasi-Monte Carlo points, $\Phi^{-1}(\mathbf{u}^{(r,n)})$ are then substituted in Eq. (5), and used as the random standard normal distributed $\xi_n(\omega)$ in order to generate random field instances.

3 Incorporating the Uncertainty in the Model

In this section, we will discuss the mechanics behind the three approaches. We will show how the evaluation points of the random field are selected on each level $\ell = \{0, \ldots, \mathrm{L}\}$ for the different approaches. Each of the approaches selects the evaluation points differently. However, all approaches start from the given set of the quadrature points $\mathbf{q}_\ell$. The points $\mathbf{q}_\ell^i \in \mathbf{q}_\ell$ are represented by $\triangle_\ell$, on a reference triangular finite element on level ℓ. Given the sets of quadrature points, the evaluation points of the random field, represented by $\bullet_\ell$, are selected on a reference triangular finite element on level ℓ, and grouped in the set $\mathbf{x}_\ell^{\text{local}}$.

The quadrature points consist of a combination of points developed by Dunavant [5] and Wandzurat [16], see Table 1. The code used to generate the Wandzurat points can be found at [3].

3.1 Non-nested Approach

For the Non-Nested Approach, the quadrature points on each level are selected as the evaluation points of the random field, i.e., $\mathbf{x}_\ell^{\text{local}} = \mathbf{q}_\ell$ for $\ell = \{0, \ldots, \mathrm{L}\}$. Because the sets of quadrature points are not nested over the levels, i.e., $\mathbf{q}_0 \nsubseteq \mathbf{q}_1 \nsubseteq \cdots \nsubseteq \mathbf{q}_\mathrm{L}$, it follows that the sets of the evaluation points of the random field are not nested, i.e., $\mathbf{x}_0^{\text{local}} \nsubseteq \mathbf{x}_1^{\text{local}} \nsubseteq \cdots \nsubseteq \mathbf{x}_\mathrm{L}^{\text{local}}$, and thus $\mathbf{x}_0 \nsubseteq \mathbf{x}_1 \nsubseteq \cdots \nsubseteq \mathbf{x}_\mathrm{L}$. This approach is the most straightforward one, and is illustrated in Fig. 2. In Algorithm 1, we present the

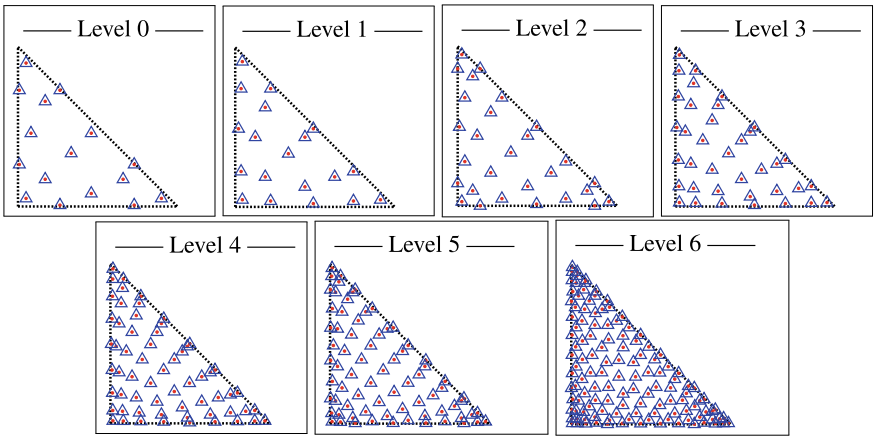


Fig. 2 Locations of the quadrature points △ and of the evaluation points of the random field ● on a reference triangular element in NNA

procedure which selects the evaluation points of the random field for each level in local coordinates and groups them in sets $\{\mathbf{x}_\ell^{\text{local}}\}_{\ell=0}^{\text{L}}$.

Algorithm 1: Generation of the evaluation points of the random field in NNA.

Data:
Max level L, Set of quadrature points per level $\{\mathbf{q}_\ell\}_{\ell=0}^{\text{L}}$
$\ell \leftarrow \text{L}$;
while $\ell \geq 0$ **do**
  $\mathbf{x}_\ell^{\text{local}} \leftarrow \mathbf{q}_\ell$;
  $\ell \leftarrow \ell - 1$;
end
return $\{\mathbf{x}_\ell^{\text{local}}\}_{\ell=0}^{\text{L}}$

3.2 Global Nested Approach

For the Global Nested Approach, we proceed in a different way. All the levels are correlated with each other. The sets of evaluation points of the random field are chosen such that they are nested over all the levels, i.e., $\mathbf{x}_0^{\text{local}} \subseteq \mathbf{x}_1^{\text{local}} \subseteq \cdots \subseteq \mathbf{x}_{\text{L}}^{\text{local}}$, and thus $\mathbf{x}_0 \subseteq \mathbf{x}_1 \subseteq \cdots \subseteq \mathbf{x}_{\text{L}}$. For GNA, the sets of evaluation points are not equal to the sets of quadrature points, except on the finest level, i.e., $\mathbf{x}_\ell^{\text{local}} \neq \mathbf{q}_\ell$ for $\ell = \{0, \ldots, \text{L} - 1\}$ and $\mathbf{x}_{\text{L}}^{\text{local}} = \mathbf{q}_{\text{L}}$. The approach for selecting the evaluation points of the random field is as follows. The quadrature points on the finest level L are selected as the evaluation

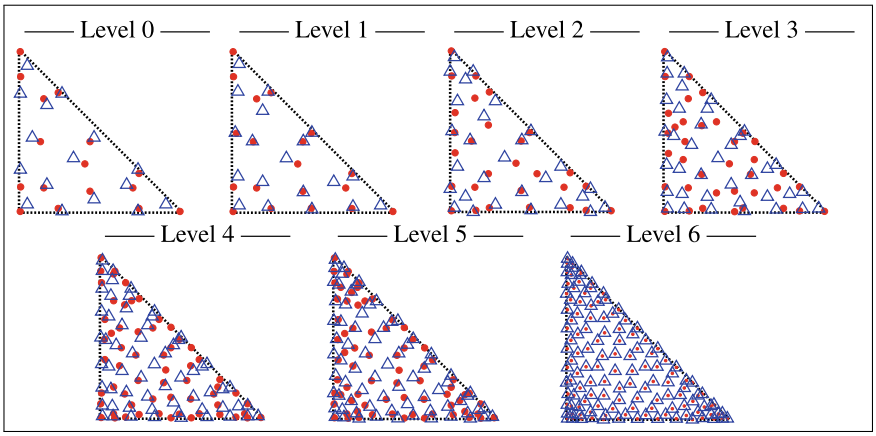


Fig. 3 Locations of the quadrature points △ and of the evaluation points of the random field ● on a reference triangular element in GNA

points of the random field, i.e., $\mathbf{x}_{\mathrm{L}}^{\mathrm{local}} = \mathbf{q}_{\mathrm{L}}$. The points selected for $\mathbf{x}_{\ell}^{\mathrm{local}}$ on levels $\ell = \{\mathrm{L}-1, \ldots, 0\}$, consist of a number of points $|\mathbf{q}_{\ell}|$, which are selected from the set $\mathbf{x}_{\ell+1}^{\mathrm{local}}$, such that each selected point is the closest neighbor of a point of the set $\mathbf{q}_{\ell}$, i.e., $\mathbf{x}_{\ell}^{\mathrm{local}} := \underset{\substack{\mathbf{x}_{\ell}^{\mathrm{local}} \subseteq \mathbf{x}_{\ell+1}^{\mathrm{local}} \\ |\mathbf{x}_{\ell}^{\mathrm{local}}| = |\mathbf{q}_{\ell}|}}{\operatorname{argmin}} \mathbf{D}\left(\mathbf{x}_{\ell+1}^{\mathrm{local}}, \mathbf{q}_{\ell}\right)$, where $\mathbf{D}(\mathbf{a}, \mathbf{b}) := \sum_{a \in \mathbf{a}} \mathbf{d}(a, \mathbf{b})$ is the distance between two sets, and where $\mathbf{d}(a, \mathbf{b}) := \inf\{\mathrm{d}(a, b) \,|b \in \mathbf{b}\}$ is the minimal distance between a point and a set, with $\mathrm{d}(a, b)$ the Euclidean distance between two points. This is illustrated in Fig. 3. The procedure used to select the evaluation points for GNA is given in Algorithm 2.

Algorithm 2: Generation of the evaluation points of the random field in GNA.

Data:
Max level L, Set of quadrature points per level $\{\mathbf{q}_{\ell}\}_{\ell=0}^{\mathrm{L}}$
$\mathbf{x}_{\mathrm{L}}^{\mathrm{local}} \leftarrow \mathbf{q}_{\mathrm{L}}$;
$\ell \leftarrow \mathrm{L}-1$;
while $\ell \geq 0$ **do**
  $i \leftarrow 1$;
  $\mathbf{x}_{\ell}^{\mathrm{local}} \leftarrow \varnothing$;
  while $i \leq |\mathbf{q}_{\ell}|$ **do**
    Find the point $p \in \mathbf{x}_{\ell+1}^{\mathrm{local}}$,which is not in $\mathbf{x}_{\ell}^{\mathrm{local}}$, closest to $\mathbf{q}_{\ell}^{i}$;
    $\mathbf{x}_{\ell}^{\mathrm{local}} \leftarrow \mathbf{x}_{\ell}^{\mathrm{local}} \cup \{p\}$; // Add it to the array
    $i \leftarrow i + 1$;
  end
  $\ell \leftarrow \ell - 1$;
end
return $\left\{\mathbf{x}_{\ell}^{\mathrm{local}}\right\}_{\ell=0}^{\mathrm{L}}$

3.3 Local Nested Approach

As was the case for the previous approaches, the user first defines which sets of quadrature points $\mathbf{q}_\ell$ are to be used. Here we set $\mathbf{q}_{\ell,\text{fine}} := \mathbf{q}_\ell$, and $\mathbf{q}_{\ell,\text{coarse}} := \mathbf{q}_{\ell-1,\text{fine}}$. The Local Nested Approach is as follows. Rather than correlating all the levels with each other, we now correlate them two-by-two. Each level $\ell = \{1, \ldots, \text{L}\}$ has two sets of evaluation points $\mathbf{x}_{\ell,\text{coarse}}$ and $\mathbf{x}_{\ell,\text{fine}}$, which are nested, i.e., $\mathbf{x}_{\ell,\text{coarse}} \subseteq \mathbf{x}_{\ell,\text{fine}}$. The points in these sets are used to generate a coarse and a fine representation of the random field on level ℓ. NNA and GNA have only one set of points per level, and thus only one representation of the random field per level. The coarse representation of the random field essentially acts as a representation of the field on level $\ell - 1$. This is because $\mathbf{q}_{\ell,\text{coarse}} = \mathbf{q}_{\ell-1,\text{fine}}$. The selection process is as follows. For each level $\ell = \{0, \ldots, \text{L}\}$, $\mathbf{x}^{\text{local}}_{\ell,\text{fine}} = \mathbf{q}_{\ell,\text{fine}}$. The points in $\mathbf{x}^{\text{local}}_{\ell,\text{coarse}}$ are selected according to the same methodology as in GNA, i.e., they are selected from the set $\mathbf{x}^{\text{local}}_{\ell,\text{fine}}$, such that each selected point is the closest neighbor to a point of the set $\mathbf{q}_{\ell,\text{coarse}}$. This is illustrated in Fig. 4. The main advantage of this approach is level exchangeability and extensibility. With exchangeability we mean that if one pair of correlated levels, say τ and $\tau - 1$, exhibits a 'sub-optimal' value of $\mathbb{V}[\Delta P_\tau]$ with respect to the variances $\mathbb{V}[\Delta P_\ell]$ on other levels, this pair can easily be exchanged against another newly computed pair with a different set of quadrature points. This is in contrast with GNA where the whole hierarchy needs to be recomputed. With level extensibility we mean that if for a user requested tolerance ε and maximum level L, the tolerance is not reached, the hierarchy can easily be extended by supplying the extra needed level(s) and reusing the previously computed samples. In case of GNA, the whole hierarchy needs to be recomputed with extra level(s) and the previously computed samples cannot be reused. This level extensibility is the major advantage of LNA over GNA.

An important note must be made concerning the LNA approach. While it successfully correlates the solutions of two successive levels, the expected value obtained from the telescoping sum is biased. We have observed a small bias of the order of 10^{-6} with respect to the actual values, an error that is well below the discretization error of the finite element discretization. The reasons behind this additional bias stems from the fact that substitute random fields are used. We are currently investigating how this additional bias can be avoided.

The procedure used to generate the point set for LNA is given in Algorithm 3. For LNA, each level has two representations of the random field, a coarse and a fine one, with sets $\mathbf{x}_{\ell,\text{coarse}}$ and $\mathbf{x}_{\ell,\text{fine}}$.

3.4 Discussion of the Computational Cost

We will now discuss how the computational cost for each of the approaches can be determined. This is done regardless of the number of samples. The total computational cost is split in an offline part, i.e., the cost of computing the eigenvalues

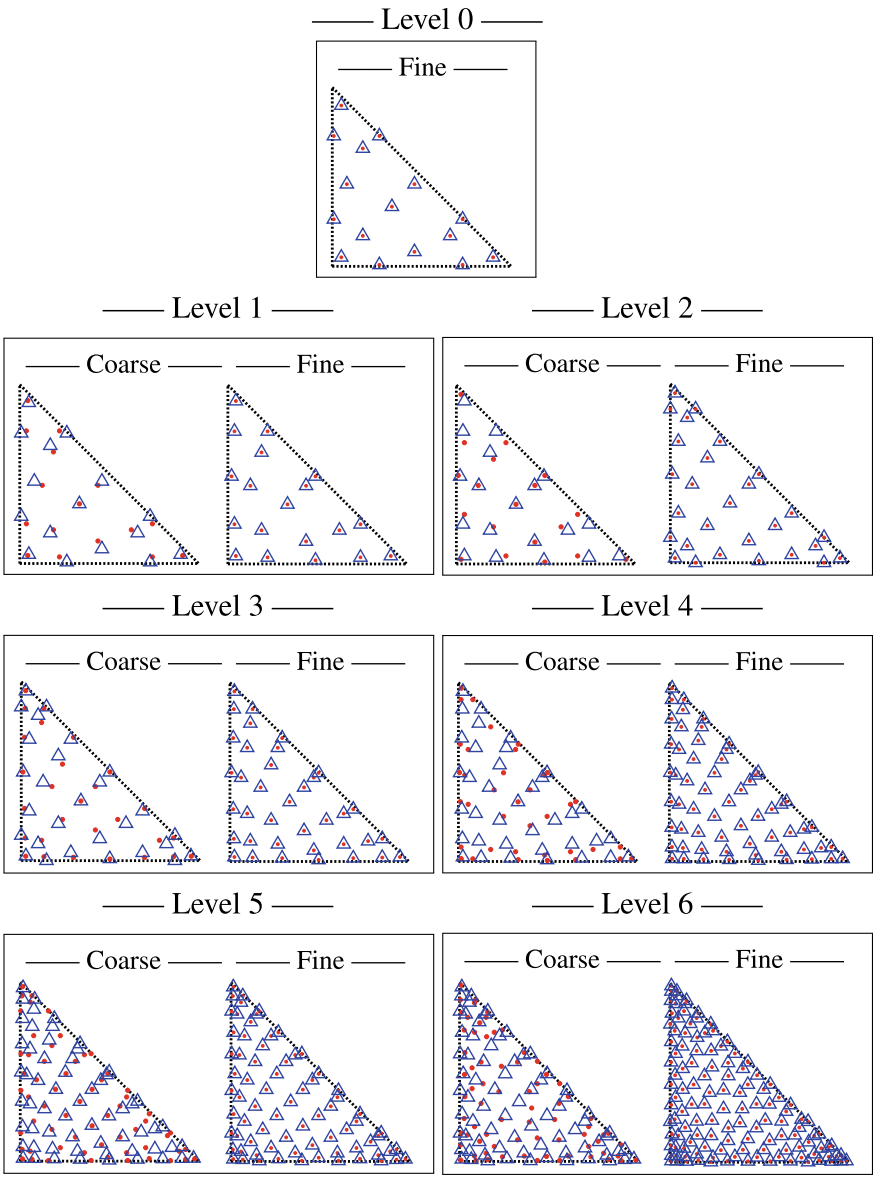


Fig. 4 Locations of the quadrature points △ and of the evaluation points of the random field ● on a reference triangular element in LNA

Algorithm 3: Generation of the evaluation points of the random field in LNA.

Data:
Max level L, Set of quadrature points per level $\left\{\mathbf{q}_{\ell,\text{coarse}}\right\}_{\ell=0}^{\text{L}}$ and $\left\{\mathbf{q}_{\ell,\text{fine}}\right\}_{\ell=0}^{\text{L}}$
$\ell \leftarrow \text{L}$;
while $\ell > 0$ **do**
  $\mathbf{x}_{\ell,\text{fine}}^{\text{local}} \leftarrow \mathbf{q}_{\ell,\text{fine}}$;
  $\mathbf{x}_{\ell,\text{coarse}}^{\text{local}} \leftarrow \varnothing$;
  $i \leftarrow 1$;
  $k \leftarrow |\mathbf{q}_{\ell,\text{coarse}}|$;
  while $i \leq k$ **do**
    Find the point $p \in \mathbf{x}_{\ell,\text{fine}}^{\text{local}}$,which is not in $\mathbf{x}_{\ell,\text{coarse}}^{\text{local}}$, closest to $\mathbf{q}_{\ell,\text{coarse}}^{i}$;
    $\mathbf{x}_{\ell,\text{coarse}}^{\text{local}} \leftarrow \mathbf{x}_{\ell,\text{coarse}}^{\text{local}} \cup \{p\}$; // Add it to the array
    $i \leftarrow i + 1$;
  end
  $\ell \leftarrow \ell - 1$;
end
$\mathbf{x}_{0,\text{fine}}^{\text{local}} \leftarrow \mathbf{q}_{0,\text{fine}}$;
return $\left\{\mathbf{x}_{0,\text{fine}}^{\text{local}}, \mathbf{x}_{1,\text{fine}}^{\text{local}}, \mathbf{x}_{1,\text{coarse}}^{\text{local}}, \mathbf{x}_{2,\text{fine}}^{\text{local}}, \mathbf{x}_{2,\text{coarse}}^{\text{local}}, \dots\right\}$

and eigenvectors of the random fields, and an online part, i.e., the cost of computing point evaluations of the random fields at the evaluation points. The offline cost is only accounted for at startup, while the online cost is accounted for when computing each sample. For NNA, the offline cost is equal to $\sum_{\ell=0}^{\text{L}} C_{\ell}^{\text{eig}}$, with C_{ℓ}^{eig} the cost of computing the eigenvalues and eigenvectors of the random field on level ℓ. For LNA, this cost is equal to $\sum_{\ell=0}^{\text{L}} C_{\ell,\text{fine}}^{\text{eig}}$. Only the eigenvalues and eigenvectors of the fine representation of the fields on each level need be computed. Note that $C_{\ell}^{\text{eig}} = C_{\ell,\text{fine}}^{\text{eig}}$. For GNA, the offline cost equals $\sum_{\ell=0}^{\text{L}} C_{\ell}^{\text{eig}}$. In case of GNA, the choice could be made to compute the eigenvectors and eigenfunctions only on the finest level L, resulting in the offline cost $C_{\text{L}}^{\text{eig}}$. This is only possible because of the property $\mathbf{x}_0 \subseteq \mathbf{x}_{\ell} \subseteq \cdots \subseteq \mathbf{x}_{\text{L}}$, see Sect. 3.2. In practice this will not be done, because of a drastic increase of the online cost, see further on. The online cost for NNA on levels $\ell > 0$ is equal to the cost of computing point evaluations of the random field on level ℓ, C_{ℓ}^{samp}, and on level $\ell - 1$, $C_{\ell-1}^{\text{samp}}$. This yields a total cost per sample equal to $C_{\ell}^{\text{samp}} + C_{\ell-1}^{\text{samp}}$. For LNA, the online cost on level $\ell > 0$ is only equal to the cost of computing point evaluations of the fine representation of the field on level ℓ, $C_{\ell,\text{fine}}^{\text{samp}}$. In order to represent the coarse field on level ℓ, a restriction of the point evaluations of the fine random field on level ℓ is taken. This is because of the property $\mathbf{x}_{\ell,\text{coarse}} \subseteq \mathbf{x}_{\ell,\text{fine}}$, see Fig. 4. There is no cost associated with the restriction. Note that $C_{\ell}^{\text{samp}} = C_{\ell,\text{fine}}^{\text{samp}}$. For GNA, the online cost on level $\ell > 0$ amounts to C_{ℓ}^{samp}, if the eigenfunctions and eigenvectors have been computed for each level ℓ. Then, the online cost is equal to the one of LNA. However, if the eigenfunctions and eigenvalues have only been computed on the finest level, the online cost for GNA equals

$C_{\mathrm{L}}^{\mathrm{samp}}$ regardless of the level. Point evaluations of the random field are computed on level L and restricted to the desired level ℓ. In practice this is not done because of the much higher cost this incurs, $C_{\mathrm{L}}^{\mathrm{samp}} \gg C_{\ell}^{\mathrm{samp}}$.

4 Model Problem

The model problem we consider for benchmarking the three approaches consists of a slope stability problem where the soil's cohesion has a spatially varying uncertainty [17]. In a slope stability problem the safety of the slope can be assessed by evaluating the vertical displacement of the top of the slope when sustaining its own weight. We consider the displacement in the plastic domain, which is governed by the Drucker–Prager yield criterion. A small amount of isotropic linear hardening is taken into account for numerical stability reasons. Because of the nonlinear stress-strain relation arising in the plastic domain, a Newton–Raphson iterative solver is used. In order to compute the displacement in a slope stability problem, an incremental load approach is used, i.e., the total load resulting from the slope's weight is added in steps starting with a force of 0 N until the downward force resulting from the slope's weight is reached. This approach results in the following system of equations for the displacement,

$$\mathbf{K}\Delta\mathbf{u} = \mathbf{r}, \tag{8}$$

where $\Delta\mathbf{u}$ stands for the displacement increment, $\mathbf{K}$ the global stiffness matrix resulting from the assembly of element stiffness matrices $\mathbf{K^e}$, see Eq. (3). The vector $\mathbf{r}$ is the residual,

$$\mathbf{r} = \mathbf{f} + \Delta\mathbf{f} - \mathbf{k}, \tag{9}$$

where $\mathbf{f}$ stands for the sum of the external force increments applied in the previous steps, $\Delta\mathbf{f}$ for the applied load increment of the current step and $\mathbf{k}$ for the internal force resulting from the stresses. For a more thorough explanation on the methods used to solve the slope stability problem we refer to [2, Chap. 2 Sect. 4 and Chap. 7 Sects. 3 and 4].

The mesh hierarchy shown in Fig. 1 is generated by using a combination of the open source mesh generator GMSH [6] and MATLAB [12]. Table 1 lists the number of elements (Nel), degrees of freedom (DOF), element order per level (Order), the number of quadrature points per element (Nquad), and the reference for the quadrature points (Ref) per level for p-MLQMC. The number of quadrature points is chosen as to increase the spatial resolution of the field per increasing level, and to ensure numerical stability of the computations of the displacement in the plastic domain. In this paper we consider two-dimensional uniform, Lagrange triangular elements.

We consider the vertical displacement in meters of the upper left node of the model as a quantity of interest (QoI). This is depicted in Fig. 5 by the arrow. The uncertainty of the soil's cohesion is represented by means of a lognormal random field. This field is obtained by applying the exponential to the field obtained in Eq. (5),

Table 1 Number of elements, degrees of freedom, element order and number of quadrature points for the model problem

p-MLQMC					
Level	Nel	DOF	Order	Nquad	Refs.
0	33	160	2	16	Dunavant [5]
1	33	338	3	19	Dunavant [5]
2	33	582	4	28	Dunavant [5]
3	33	892	5	37	Dunavant [5]
4	33	1268	6	61	Dunavant [5]
5	33	1710	7	73	Dunavant [5]
6	33	2218	8	126	Wandzurat and Xiao [16]

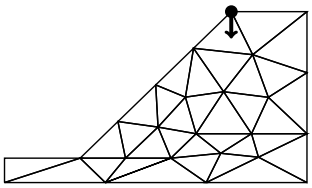


Fig. 5 The vertical displacement of the upper left node as the QoI, indicated with an arrow

$Z_{\text{lognormal}}(\mathbf{x}, \omega) = \exp(Z(\mathbf{x}, \omega))$. For the covariance kernel of the random field, we use the Matérn covariance kernel,

$$C(\mathbf{x}, \mathbf{y}) := \sigma^2 \frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{y}\|_2}{\lambda} \right)^{\nu} K_{\nu} \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{y}\|_2}{\lambda} \right), \tag{10}$$

with $\nu = 2.0$ the smoothness parameter, K_{ν} the modified Bessel function of the second kind, $\sigma^2 = 1$ the variance and $\lambda = 0.3$ the correlation length. The characteristics of the lognormal distribution used to represent the uncertainty of the soil's cohesion are as follows: a mean of 8.02 kPa and a standard deviation of 400 Pa. The spatial dimensions of the slope are: a length of 20 m, a height of 14 m and a slope angle of 30°. The material characteristics are: a Young's modulus of 30 MPa, a Poisson ratio of 0.25, a density of 1330 kg/m^3 and a friction angle of 20°. Plane strain is considered for this problem. The number of stochastic dimensions considered for the generation of the Gaussian random field is $s = 400$, see Eq. (5). With a value $s = 400$, 99% of the variability of the random field is accounted for.

The stochastic part of our simulations was performed with the Julia packages **MultilevelEstimators.jl**, see [15], and **GaussianRandomFields.jl**, see [14]. The finite element code used is an in-house MATLAB code developed by the Structural Mechanics Section of the KU Leuven. All the results have been computed on a workstation equipped with 2 physical cores, Intel Xeon E5-2680 v3 CPU's, each with 12 logical cores, clocked at 2.50 GHz, and a total of 128 GB RAM.

5 Numerical Results

In this section we present our numerical results obtained with the p-MLQMC method.

5.1 Displacement

In Fig. 6, we show the displaced meshes and their nodes for a single sample of the random field on different levels. For better visualization, the displacement of the mesh and nodes in the figure have been exaggerated by a factor 20. The value of the QoI is listed beneath each figure depicting the displacement.

5.2 Variance and Expected Value Over the Levels

In Fig. 7 we show the sample variance over the levels $\mathbb{V}[P_\ell]$, the sample variance of the difference over the levels $\mathbb{V}[\Delta P_\ell]$, the expected value over the levels $\mathbb{E}[P_\ell]$ and the expected value of the difference over the levels $\mathbb{E}[\Delta P_\ell]$.

As expected with multilevel methods, we observe that $\mathbb{E}[P_\ell]$ remains constant over the levels, while $\mathbb{E}[\Delta P_\ell]$ decreases with increasing level. This is the case for all approaches we introduced in Sect. 3.

As explained in Sect. 2, multilevel methods are based on a variance reduction by means of a hierarchical refinement of finite element meshes. In practice this means that the sample variance $\mathbb{V}[P_\ell]$ remains constant across the levels, while the sample variance of the difference over the levels $\mathbb{V}[\Delta P_\ell]$ decreases for increasing level. This

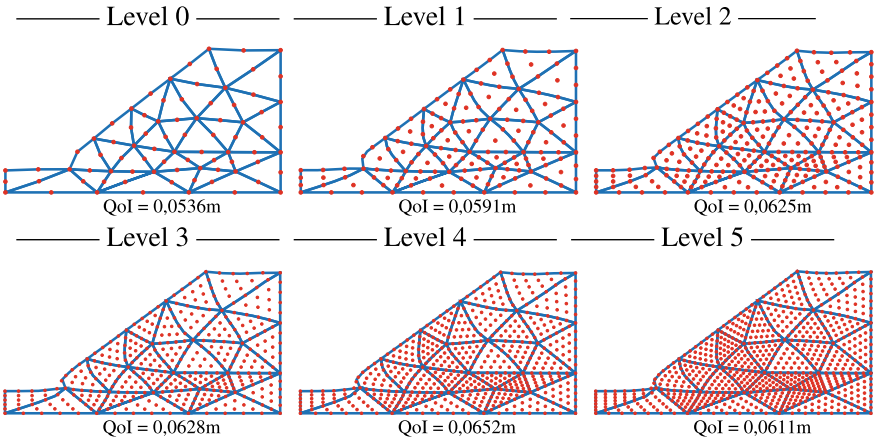


Fig. 6 Displaced meshes and QoI for different samples of the random field

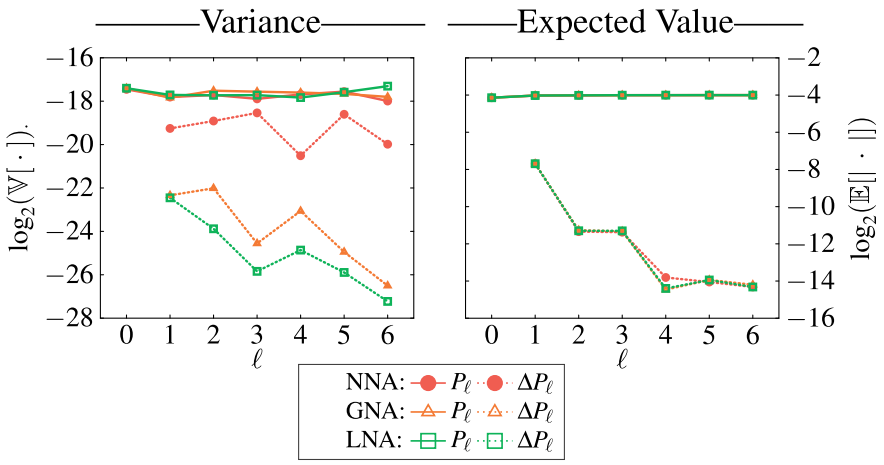


Fig. 7 Variance and Expected Value over the levels

is indeed what we observe for GNA and LNA. For NNA we observe that $\mathbb{V}[\Delta P_\ell]$ does not decrease. From Fig. 7, we can conclude that the choice of using nested over non-nested spatial locations over the levels as evaluation points for the random field greatly improves the behavior of $\mathbb{V}[\Delta P_\ell]$. This influence stems from a 'bad' correlation between the results of two successive levels in the NNA case, see Eq. (2). We will show in the next section that the number of samples per level required by NNA will be larger than GNA or LNA. This will impact the total runtime.

5.3 Number of Samples

In Fig. 8, we show the number of samples for the three approaches for thirteen different tolerances on the RMSE. These numbers do not include the number of shifts, which value is taken to be $R_\ell = 10$ for $\ell = \{0, \ldots, \mathrm{L}\}$.

We observe that for a given tolerance ε, the number of samples for NNA for levels greater than 0 is higher than for GNA and LNA. This is due to the slow decrease of $\mathbb{V}[\Delta P_\ell]$, see Fig. 7.

Unlike Multilevel Monte Carlo, the number of sample per level for MLQMC is not the result of an optimization problem, see [7]. The number of samples in MLQMC are chosen according to an adaptive 'doubling' algorithm, where the number of samples is each time multiplied by a factor until the statistical constrain are satisfied. For a more thorough explanation we refer to [9, Sect. 5]. The advantage of an adaptive algorithm consists of the fact that no equation expressing the number of samples in function of the variance has to be derived. Indeed, such an expression would require the evaluation of terms which are difficult to estimate at runtime. In our approach, the sample multiplication factor is chosen as 1.2. The multiplication factor of 1.2 is

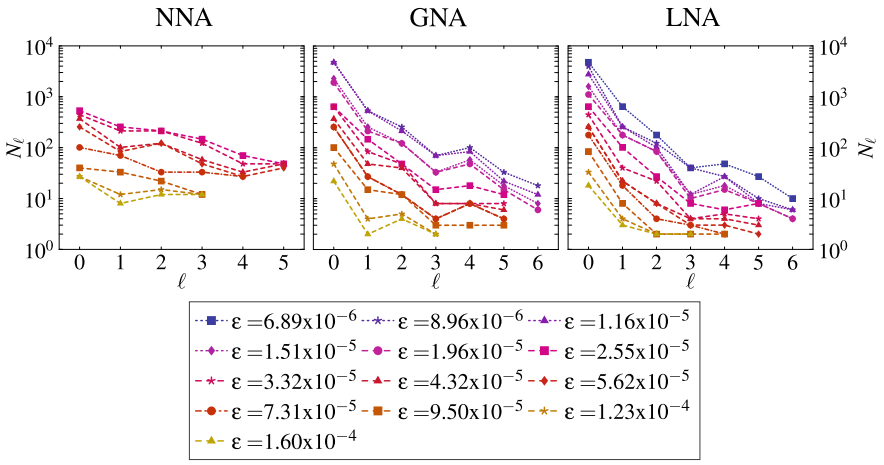


Fig. 8 Number of samples for the three implementations for a given tolerance ε

chosen over the more naive value of 2 because, in the current setting each sample involves the solution of a complex PDE, where the doubling of the number of samples would lead to huge jumps in the total cost of the estimator. The reduction from 2 to 1.2 is a compromise leading to a more gradual increase of the computational cost, while ensuring that enough progress is made.

5.4 Runtimes

We show the absolute and relative runtimes as a function of the user requested tolerance ε on the RMSE for the different implementations in Fig. 9.

For the shown results, the maximal number of available levels in the p-MLQMC method has been set to 7, i.e., L = 6. The algorithm adaptively choses the number of levels needed to satisfy a given tolerance on the RMSE during runtime. If for a given tolerance on the RMSE, 8 levels are needed, an extra level can easily be added in case of the LNA approach while reusing the previously computed samples. For GNA, an extra level can also be added but the previously computed samples can not be reused in the new hierarchy of meshes.

The results for the absolute runtime are expressed in seconds. For the relative runtime, we have normalized the computational cost of all three approaches such that the results for LNA for each tolerance have unity cost. For both the results of the absolute and the relative runtime, we show the average, computed over three independent simulation runs, together with the minimum and maximum bounds.

We observe that for a given tolerance, LNA achieves a speedup up to a factor 5 with respect to NNA and a factor 1.5 with respect to GNA. GNA also outperforms NNA in terms of lower computational cost. This better performance of GNA and LNA is due

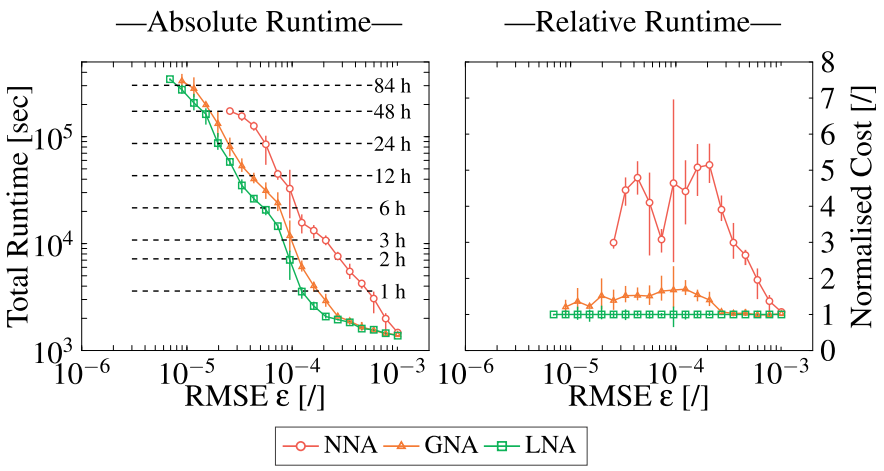


Fig. 9 Runtimes as a function of requested user tolerance

to a lower number of samples per level, resulting from a better correlation between the successive levels. We can thus state that the Local and Global Nested Approaches achieve a lower computational cost with respect to the Non-Nested Approach for a given tolerance.

6 Conclusions

In this work, we investigated how the spatial locations used for the evaluation of the random field by means of a Karhunen-Loève expansion impact the performance of the p-MLQMC method. We distinguished three different approaches, the *Non-Nested Approach*, the *Global Nested Approach* and the *Local Nested Approach*. We demonstrated that the choice of the evaluation points of the random field impacts the variance reduction over the levels $\mathbb{V}[\Delta P_\ell]$. We showed that the Global and Local Nested approaches exhibit a much better decrease of $\mathbb{V}[\Delta P_\ell]$ due to a better correlation between the levels than the Non-Nested Approach. This leads to a lower number of samples for the Nested Approaches and thus a lower total runtime for a given tolerance. Furthermore we have shown that the Local Nested Approach has the additional properties of level exchangeability and extensibility with respect to the Global Nested Approach. By correlating the levels two-by-two, in the Local Nested Approach, one pair of levels can easily be exchanged for another computed pair, if needed. In addition, the hierarchy can also easily be extended by adding a newly computed pair. When exchanging a pair of levels or extending the hierarchy, the previously computed samples can be reused. The Global Nested Approach does not have these properties. There, the whole mesh hierarchy needs to be recomputed with the extra added and/or exchanged level(s). The previously computed samples

cannot be reused. In addition to these properties, the Local Nested Approach also has a smaller runtime for a given tolerance on the RMSE than the Global Nested Approach. Based on the results in this work, we conclude that for selecting the random field evaluation points, an approach where the points are nested across the mesh hierarchy provides superior results compared to a non-nested approach. Of the nested approaches we consider in this paper, the Local Nested Approach has the smallest computational runtime, outperforming the Non-Nested Approach by a factor 5.

Acknowledgements The authors gratefully acknowledge the support from the Research Council of KU Leuven through project C16/17/008 "Efficient methods for large-scale PDE-constrained optimization in the presence of uncertainty and complex technological constraints". The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation—Flanders (FWO) and the Flemish Government—department EWI.

References

1. Blondeel, P., Robbe, P., Van hoorickx, C., François, S., Lombaert, G., Vandewalle, S.: p-refined multilevel Quasi-Monte Carlo for Galerkin finite element methods with applications in civil engineering. Algorithms **13**(5) (2020). https://doi.org/10.3390/a13050110, https://www.mdpi.com/1999-4893/13/5/110
2. de Borst, R., Crisfield, M.A., Remmers, J.J.C.: Non Linear Finite Element Analysis of Solids and Structures. Wiley, U.K. (2012)
3. Burkardt, J.: Matlab implementation of Wandzura rule (2007). Online https://people.math.sc.edu/Burkardt/m_src/triangle_wandzura_rule/triangle_wandzura_rule.html. Accessed 01 Oct 2020
4. Cliffe, K.A., Giles, M.B., Scheichl, R., Teckentrup, A.L.: Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients. Comput. Vis. Sci. **14**(1), 3 (2011). https://doi.org/10.1007/s00791-011-0160-x
5. Dunavant, D.A.: High degree efficient symmetrical Gaussian quadrature rules for the triangle. Int. J. Numer. Meth. Eng. **21**(6), 1129–1148 (1985). https://doi.org/10.1002/nme.1620210612
6. Geuzaine, C., Remacle, J.F.: Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Numer. Meth. Eng. **79**(11), 1309–1331 (2009). https://doi.org/10.1002/nme.2579
7. Giles, M.B.: Multilevel Monte Carlo path simulation. Oper. Res. **56**(3), 607–617 (2008). https://doi.org/10.1287/opre.1070.0496
8. Giles, M.B.: Multilevel Monte Carlo methods. Acta Num. **24**, 259–328 (2015). https://doi.org/10.1017/S096249291500001X
9. Giles, M.B., Waterhouse, B.J.: Multilevel Quasi-Monte Carlo path simulation. Rad. Ser. Comput. App. **8**, 1–18 (2009)
10. Kuo, F.: Lattice rule generating vectors (2007). Online https://web.maths.unsw.edu.au/~fkuo/lattice/index.html and https://web.maths.unsw.edu.au/~fkuo/lattice/lattice-32001-1024-1048576.3600. Accessed 12 Apr 2019
11. Li, C., Kiureghian, A.D.: Optimal discretization of random fields. J. Eng. Mech. **119**(6), 1136–1154 (1993). https://doi.org/10.1061/(ASCE)0733-9399(1993)119:6(1136)
12. MATLAB: version 9.2.0 (R2017a). The MathWorks Inc., Natick, Massachusetts (2017)
13. Matthies, H.G., Brenner, C.E., Bucher, C.G., Guedes Soares, C.: Uncertainties in probabilistic numerical analysis of structures and solids-stochastic finite elements. Struct. Saf. **19**(3), 283–336 (1997)

14. Robbe, P.: Gaussianrandomfields.jl (2017). Online https://github.com/PieterjanRobbe/GaussianRandomFields.jl. Accessed 05 Nov 2020
15. Robbe, P.: Multilevelestimators.jl (2018). Online https://github.com/PieterjanRobbe/MultilevelEstimators.jl. Accessed 05 Nov 2020
16. Wandzurat, S., Xiao, H.: Symmetric quadrature rules on a triangle. Comput. Math. Appl. **45**(12), 1829–1840 (2003). https://doi.org/10.1016/S0898-1221(03)90004-6
17. Whenham, V., De Vos, M., Legrand, C., Charlier, R., Maertens, J., Verbrugge, J.C.: Influence of soil suction on trench stability. In: Schanz, T. (ed.) Experimental Unsaturated Soil Mechanics, pp. 495–501. Springer, Berlin (2007)

Scalable Control Variates for Monte Carlo Methods Via Stochastic Optimization

Shijing Si, Chris. J. Oates, Andrew B. Duncan, Lawrence Carin, and François-Xavier Briol

Abstract Control variates are a well-established tool to reduce the variance of Monte Carlo estimators. However, for large-scale problems including high-dimensional and large-sample settings, their advantages can be outweighed by a substantial computational cost. This paper considers control variates based on Stein operators, presenting a framework that encompasses and generalizes existing approaches that use polynomials, kernels and neural networks. A learning strategy based on minimizing a variational objective through stochastic optimization is proposed, leading to scalable and effective control variates. Novel theoretical results are presented to provide insight into the variance reduction that can be achieved, and an empirical assessment, including applications to Bayesian inference, is provided in support.

Keywords Stein's method · Variance reduction · Monte Carlo methods · Control variates

S. Si · L. Carin
Duke University, Durham, CA, USA
e-mail: shijing.si@outlook.com

L. Carin
e-mail: lcarin@duke.edu

Chris. J. Oates
Newcastle University, Newcastle upon Tyne, England
e-mail: chris.oates@ncl.ac.uk

A. B. Duncan
Imperial College London, London, England
e-mail: a.duncan@imperial.ac.uk

F.-X. Briol (✉)
University College London, London, England
e-mail: f.briol@ucl.ac.uk

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_10

1 Introduction

This paper focuses on the approximation of the integral of an arbitrary function $f : \mathbb{R}^d \to \mathbb{R}$ with respect to a distribution Π, denoted $\Pi[f] := \int f \mathrm{d}\Pi$. It will be assumed that Π admits a smooth and everywhere positive Lebesgue density π such that the gradient of $\log \pi$ can be pointwise evaluated. This situation is typical in Bayesian statistics, where Π represents a posterior distribution and, to circumvent this intractability, Markov chain Monte Carlo (MCMC) methods are used. Nevertheless, the ergodic average of MCMC output converges at a slow rate proportional to $n^{-1/2}$ and, for finite chain length n, there can be considerable stochasticity associated with the MCMC output.

A control variate (CV) is a variance reduction technique for Monte Carlo (MC) methods, including MCMC. Given a test function f, the general approach is to identify another function, g, such that the variance of the estimator with f replaced by $f - g$ is smaller than that of the original estimator, and such that $\Pi[g] = 0$, so the value of the integral is unchanged. Such a g is called a CV. CVs are widely-used in statistics and machine learning, including for the simulation of Markov processes [25, 37], stochastic optimization [55], stochastic gradient MCMC [3], reinforcement learning [22, 23, 30], variational inference [42, 46, 47] and Bayesian evidence evaluation [40].

Given a test function f, the problem of selecting an appropriate CV is non-trivial and a variety of approaches have been proposed. Our discussion focuses only on the setting where π is provided only up to an unknown normalization constant; i.e., the setting where MCMC is typically used. The most widely-used approach to selection of a CV is based on $g = \nabla \log \pi$ and simple (e.g., linear) transformations thereof [2, 17, 35, 43]; note that under weak tail conditions on π, the CV property $\Pi[g] = 0$ is assured. Recently several authors have proposed the use of more complicated or even non-parametric transformations, such as based on high order polynomials [52], kernels [6, 38, 39] and neural networks (NNs) [22, 30, 36, 54]. These new approaches have been shown empirically—and theoretically, in the case of kernels [6, 38]—to provide substantial reduction in variance for MCMC.

These recent developments are closely related to Stein's method [12, 49, 53], a tool used in probability theory to quantify how well one distribution Π' approximates another distribution Π. Recall that, given a collection of functions g for which $\Pi[g] = 0$ is satisfied, Stein's method uses $\sup_g \Pi'[g]$ as a means of quantifying the difference between Π and Π'. As a byproduct, researchers in this field have constructed a large range of functions g that can be used as CVs. Although Stein's method has recently been applied to a variety of problems including MCMC convergence assessment [19–21], goodness-of-fit testing [15, 32, 56], variational inference [46, 47], estimators for models with intractable likelihoods [5, 34] and the approximation of complex posterior distributions [13, 14, 31–33, 48], a unified account of how Stein's method can be exploited for the construction of CVs, encompassing existing polynomial, kernel and NN transformations, has yet to appear.

The organization and contributions of this paper are as follows. The literature on polynomial, kernel, and NN CVs is reviewed in Sect. 2. An efficient learning strategy for CVs based on stochastic optimization is proposed in Sect. 3. A theoretical analysis is provided in Sect. 4, which provides general sufficient conditions for variance reduction to be achieved. Finally, an empirical assessment is provided in Sect. 5 and covers a range of synthetic test problems, as well as problems arising in the Bayesian inferential context.

2 Background

In what follows, it is assumed that an approximate sample $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$ from Π have been obtained and our goal is to construct an estimator for $\Pi[f]$ of the form $\frac{1}{n-m}\sum_{i=m+1}^n f(x_i) - g(x_i)$ where g is a CV learned using a subset of size $m \leq n$ from the $\{x_i\}_{i=1}^n$.

Several approaches have been proposed. One approach is to use a Taylor expansion of the test function f [42, 55], or perhaps a polynomial approximation to f learned from regression [28]. Unfortunately, this will only be a feasible approach when integrating against simple probability distributions Π for which polynomials can be exactly integrated, such as a Gaussian. CVs may also be directly available through problem-specific knowledge, e.g., for certain Markov processes [25, 37], but this is rarely the case in general. Alternatively, CVs can sometimes be built using known properties of the method used for obtaining samples; see [1, 7, 9, 11, 16, 24] for CVs that are developed with a particular MCMC method in mind. See also [26] for CVs specialized to quasi Monte Carlo (QMC). An obvious drawback to the methods above is that they impose strong restrictions on the methods that one may use to obtain the $\{x_i\}_{i=1}^m$.

An arguably more general framework, and our focus in this paper, is to first curate a rich set $\mathcal{G}$ of candidate CVs, and then to employ a learning procedure to approximately select an optimal CV $g \in \mathcal{G}$. This should be done according to a suitable optimality criterion based on f and the given set $\{x_i\}_{i=1}^m$. The methodological challenges are therefore twofold; first, we must construct $\mathcal{G}$ and second, we must provide a procedure to select a suitable CV from this set. The construction of a candidate set $\mathcal{G}$ has been approached by several authors using a variety of regression-based techniques:

- Motivated by physical considerations, [2] proposed to use $g = Hu$, based on the Schrödinger-type Hamiltonian $H = -0.5\Delta + 0.5(\sqrt{\pi})^{-1}\Delta\sqrt{\pi}$, where Δ is the Laplacian and u is a polynomial of fixed degree. See also [17, 35, 43].
- An approach called *control functionals* (CFs) was proposed in [39], where the set $\mathcal{G}$ consisted of functions of the form $g = \nabla \cdot u + u \cdot \nabla \log \pi$, where $\nabla\cdot$ denotes the divergence operator, ∇ denotes the gradient operator and $u : \mathbb{R}^d \to \mathbb{R}^d$ is constrained to belong to a suitable Hilbert space of vector fields on $\mathbb{R}^d$. See also [6, 38, 51] for the connection with Stein's method.

- In more recent work, [54] extended the CF approach to the case where a NN is used to provide a parametric family of candidates for the vector field u. The set of all such functions g generated using a fixed architecture of NN is taken as $\mathcal{G}$. See also [30, 36].

Thus, several related options are available for constructing a suitable candidate set $\mathcal{G}$. However, where existing literature diverges markedly is in the procedure used to select a suitable CV from this set:

- For approaches based on polynomials, [2] proposed to select polynomial coefficients θ in order to minimize the sum-of-squares error $\sum_{i=1}^{m}(f(x_i) - g_\theta(x_i))^2$. Here g_θ is used to emphasize the dependence on coefficients θ of the polynomial. For even moderate degree polynomials, the combinatorial explosion in the number of coefficients as d grows necessitates regularized estimation of θ; suitable regularizers are evaluated in [45, 52].
- For the CF approaches, regularized estimation is essential since the Hilbert space is infinite dimensional. Here, [39] proposed to select g as a minimal norm element of the Hilbert space for which the interpolation equations $f(x_i) = c + g(x_i)$ are satisfied for all $i = 1, \ldots, m$ and some $c \in \mathbb{R}$. A major drawback of this approach is the $O(m^3)$ computational cost.
- The approach based on NN also exploited a sum-of-squares error, but in [54] the authors proposed to include an additional regularizer term $\lambda \sum_{i=1}^{m} g_\theta(x_i)^2$, for some pre-specified constant λ, to avoid over-fitting of the NN. Optimization over θ, the parameters of the NN that enter into g_θ, was performed using stochastic gradient descent.

It is therefore apparent that, in existing literature, the construction of the candidate set $\mathcal{G}$ is intimately tied to the approach used to select a suitable element from it. This makes it difficult to draw meaningful conclusions about which CVs are most suitable for a given task; from a theoretical perspective, existing analyses make assumptions that are mutually incompatible and, from a practical perspective, the different techniques and software involved in implementing existing methods precludes a straightforward empirical comparison. Our attention therefore turns next to the construction of a general framework that can be used to learn a wide range of CVs, including polynomial, kernel and NN, under a single set of theoretical assumptions and algorithmic parameters, enabling a systematic assessment of CV methods to be performed.

3 Methods

Here we present a general framework for the construction of CVs: In 3.1 the construction of a candidate set $\mathcal{G}$ is achieved using Stein operators, which unifies the CVs proposed in existing contributions such as [2, 39, 54] and covers simultaneously the case of polynomials, kernels and NNs. Then, in 3.2, we present an approach to selec-

tion of a suitable element $g \in \mathcal{G}$, based on a variational formulation and performing stochastic optimization on an appropriate objective functional.

3.1 Classes of Control Variates $\mathcal{G}$

The construction of non-trivial functions $g : \mathbb{R}^d \to \mathbb{R}$ with the property $\Pi[g] = 0$ is not straight-forward in the setting where MCMC would be used, since for general f the integral $\Pi[f]$ cannot be exactly computed. Stein's method [53] offers a solution to this problem in the case where the gradient of $\log \pi$ can be evaluated pointwise, which we describe next. A *Stein characterization* of a distribution Π consists of a pair $(\mathcal{U}, \mathcal{L})$, where $\mathcal{U}$ is a set of functions whose domain is $\mathbb{R}^d$ and $\mathcal{L}$ is an operator, such that $\Pi'[\mathcal{L}u] = 0 \; \forall u \in \mathcal{U}$ if and only if the distributions Π' and Π are equal. In this case $\mathcal{U}$ is called a *Stein class* and $\mathcal{L}$ is called a *Stein operator*.[1] Clearly, if one can identify a Stein characterization for Π, then one could take $\mathcal{G} = \mathcal{L}\mathcal{U} = \{\mathcal{L}u : u \in \mathcal{U}\}$ as a set of candidates CVs.

The literature on Stein's method provides general approaches to identify a Stein characterization [12, 49]. In the *generator approach*, $\mathcal{L}$ is taken to be the infinitesimal generator of a Markov process which is ergodic with respect to Π [4]. For example, if $\mathcal{L}$ is the infinitesimal generator of an overdamped Langevin diffusion then one obtains the *Langevin* Stein operator, which acts on vector fields u on $\mathbb{R}^d$ as $\mathcal{L}_{\mathrm{L}}u = \nabla \log \pi \cdot u + \nabla \cdot u$. This recovers the operator used in the control functional (CF) approach of [39], as well as the operator used in the NN approach of [54]. Alternatively, we could construct an operator that acts on *scalar*-valued functions by replacing the vector field u with the potential ∇u in the previous operator, leading to the scalar-valued Langevin (SL) Stein operator $\mathcal{L}_{\mathrm{SL}}u = \Delta u + \nabla u \cdot \nabla \log \pi$. This recovers the operator used with polynomials in [2, 35]. Trivially, a scalar multiple of a Stein operator is a Stein operator, and one may combine Stein characterizations $(\mathcal{U}_i, \mathcal{L}_i)$ linearly as $\mathcal{L}u = \mathcal{L}_1 u_1 + \mathcal{L}_2 u_2$, $u \in \mathcal{U}_1 \times \mathcal{U}_2$, so that considerable flexibility can be achieved. We will see in Sect. 5 that this can lead to scalable and flexible classes of CVs.

3.2 Selection of a Control Variate $g \in \mathcal{G}$

Once a set $\mathcal{G}$ of candidate CVs has been constructed, we must consider how to select a suitable element $g \in \mathcal{G}$ (or equivalently $u \in \mathcal{U}$) that leads to improved performance of the MC estimator when f is replaced by $f - g$. In general this will depend on the specific details of the MC method; for example, in MCMC one would select g to minimize asymptotic variance [7, 16], while in QMC one would minimize the

[1] To simplify presentation in the paper, we always assume $\mathcal{U}$ is a *maximal* set of functions for which $\mathcal{L}u$ is well-defined and $\Pi[\mathcal{L}u] = 0$.

Hardy-Krause variation [26]. The situation simplifies considerably when $\mathcal{G}$ contains an element g^* such that $f - g^*$ is constant. This optimal function $g^* = \mathcal{L}u^*$, if it exists, is given by the solution of *Stein's equation*: $\mathcal{L}u^*(x) = f(x) - \Pi[f]$. This paper proposes to directly approximate a solution of this equation (a linear partial differential equation) by casting it in a variational form and solving over a subset $\mathcal{V} \subseteq \mathcal{U}$. The variational characterization that we use is that $J(u^*) = 0$, where

$$J(u) := \|f - \mathcal{L}u - \Pi[f]\|^2_{L^2(\Pi)} = \mathrm{Var}_\Pi[f - \mathcal{L}u],$$

with $L^2(\Pi)$ being the space of square-integrable functions with respect to Π. In the spirit of empirical risk minimization, we propose to minimize an empirical approximation of this functional, computed based on samples $(x_i)_{i=1}^m$ that are drawn either exactly or approximately from Π. There are two natural approximations that could be considered. The first is based on the *variance* representation

$$\begin{aligned} J(u) &= \mathrm{Var}_\Pi[f - \mathcal{L}u] \approx J_m^{\mathrm{V}}(u) \\ J_m^{\mathrm{V}}(u) &:= \tfrac{2}{m(m-1)} \textstyle\sum_{i>j} (f(x_i) - \mathcal{L}u(x_i) - f(x_j) + \mathcal{L}u(x_j))^2, \end{aligned} \tag{1}$$

providing an approximation of J at cost $O(m^2)$, used in [8]. The second is based on the *least-squares* representation

$$\begin{aligned} J(u) &= \min_{c\in\mathbb{R}} \|f - \mathcal{L}u - c\|^2_{L^2(\Pi)} \approx \min_{c\in\mathbb{R}} J_m^{\mathrm{LS}}(c, u), \\ J_m^{\mathrm{LS}}(c, u) &:= \tfrac{1}{m} \textstyle\sum_{i=1}^m (f(x_i) - \mathcal{L}u(x_i) - c)^2, \end{aligned} \tag{2}$$

providing an approximation of J at cost $O(m)$, used in [2, 35, 38, 39]. These approximations will be unbiased when the x_i are independent draws from Π, but this will not necessarily hold for the MCMC case. To approximately solve this variational formulation we consider a parametric subset $\mathcal{V} \subseteq \mathcal{U}$, where elements of $\mathcal{V}$ can be written as v_θ for some parameter $\theta \in \mathbb{R}^p$. Depending on the specific nature of the functions g_θ, it can occur that the optimization problem is under-constrained, e.g., when $p > m$. Therefore, following [39, 52, 54], we also allow for the possibility of additional regularization at the level of θ. Thus we aim to minimize objectives of the form $\tilde{J}_m^{\mathrm{V}}(\theta) + \lambda_m \Omega(\theta)$ and $\tilde{J}_m^{\mathrm{LS}}(c, \theta) + \lambda_m \Omega(\theta)$ over $c \in \mathbb{R}$ and $\theta \in \mathbb{R}^p$, where $\tilde{J}_m^{\mathrm{V}}(\theta) := J_m^{\mathrm{V}}(v_\theta)$, $\tilde{J}_m^{\mathrm{V}}(c, \theta) := J_m^{\mathrm{V}}(c, v_\theta)$, $\lambda_m > 0$ and $\Omega(\theta)$ is a regularization term to be specified. To reduce notational overhead, for the least-squares case we let $\theta_0 := c$ and simply write $\tilde{J}_m^{\mathrm{LS}}(\theta)$ where $\theta \in \mathbb{R}^{p+1}$.

To perform the minimization, we propose to use stochastic gradient descent (SGD). Thus, to minimize a functional $F(\theta)$, we iterate through $\theta^{(t+1)} = \theta^{(t)} - \alpha_t \widehat{\nabla F}(\theta^{(t)})$, where the *learning rate* α_t decreases as $t \to \infty$ and $\widehat{\nabla F}$ is an unbiased approximation to ∇F. In our experiments, $\widehat{\nabla F}$ is constructed using a randomly chosen subset from $(x_i)_{i=1}^m$, with this subset being re-sampled at each step of SGD (*i.e.*, mini-batch SGD).

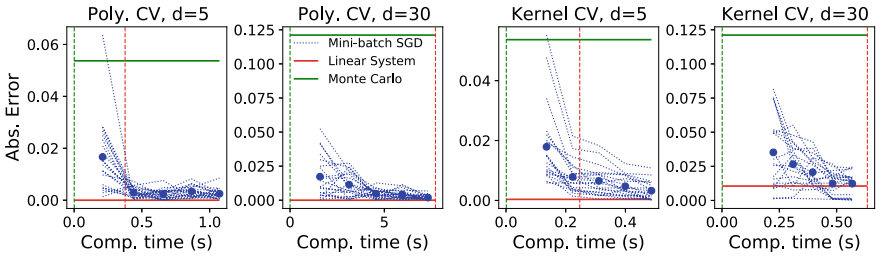


Fig. 1 *Scalable Control Variates in High Dimensions.* Here we consider the toy problem of integrating $f(x) = x_1 + \cdots + x_d$ against $\mathcal{N}(0, I_{d\times d})$. The total sample size is $n = 1000$ and $m = 500$ of these were used as the training set. Here 20 realizations (blue dashed lines) are shown and blue dots represent the mean absolute error. The red lines represent the performance and computational cost of solving the corresponding linear system exactly, our benchmark. Similarly, the green lines represent the MC estimator with no CV used

This framework is compatible with any parametric function class and has the potential to provide significant speed-ups, relative to existing methods, due to the efficiency of SGD. For example, taking $\mathcal{V}$ to be the polynomials of degree at most k in each variable recovers the same class as [2, 35, 43, 52], but with a parameter optimization strategy based on SGD as opposed to exact least squares. This problem is hence closely related to the ADALINE algorithm with basis functions which integrate to zero.

For SGD with t iterations and mini-batches of size b, our computational cost will be of order $\mathcal{O}(d^k bt)$, whereas exact least squares must solve a linear system of size $O(d^k)$, leading to a cost of $\mathcal{O}(d^{3k} + md^k)$. Similarly, taking $\mathcal{V}$ to be a linear space spanned by m translates of a kernel recovers the CF method of [39]. SGD has computational cost of $\mathcal{O}(mdbt)$, whereas CFs requires $\mathcal{O}(m^3 + m^2 d)$ due to the need to invert an m-dimensional matrix.

Significant reduction in computational cost can also be obtained for ensembles: a combination of polynomial and kernel basis functions, as considered in [51], would cost $\mathcal{O}((md + d^k)bt)$ compared to the $\mathcal{O}(m^3 + d^{3k} + m^2 + md^k)$ cost when the linear system is exactly solved. Furthermore, any hyper-parameters, such as kernel parameters, can be incorporated into the minimization procedure with SGD, so that nested computational loops are avoided.

Some of these speed-ups are illustrated on a toy example in Fig. 1. Even for this moderately-sized problem, the use of SGD provides significant speed-ups. Additional experiments with values of $m = 5000$ in [50] show that larger speed-ups can be obtained for large scale problems.

4 Theoretical Assessment

In this section we present our novel theoretical results for CVs trained using SGD. All proofs are contained in the Appendix.

The first question is whether it is possible to obtain *zero-variance* CVs, i.e. can we find a $u \in \mathcal{U}$ such that $J(u) = \mathrm{Var}_\Pi[f - \mathcal{L}u] = 0$. The answer is "yes" under regularity conditions on Π and $\mathcal{L}$, and whenever $\mathcal{V}$ is large enough. In particular, a fixed parametric class may not be large enough, but we can consider a nested sequence of sets $\mathcal{V}_1 \subseteq \mathcal{V}_2 \subseteq \cdots$ such that $\cup_{p\in\mathbb{N}}\mathcal{V}_p$ is dense in $\mathcal{U}$. For example, $\mathcal{V}_p$ could be polynomials of degree p, or NNs with p hidden units.

Proposition 1 *Let $\mathcal{U}$ be a normed space and $\mathcal{L} : \mathcal{U} \to L^2(\Pi)$ be a bounded linear operator. Consider a sequence of nested sets $\mathcal{V}_1 \subseteq \mathcal{V}_2 \subseteq \cdots$ such that $\cup_{p\in\mathbb{N}}\mathcal{V}_p$ is dense in $\mathcal{U}$. If $\exists u \in \mathcal{U}$ that solves the Stein equation $\mathcal{L}u = f - \Pi[f]$, then $\lim_{p\to\infty} \inf_{v\in\mathcal{V}_p} J(v) = 0$.*

Of course, the existence of a solution to the Stein equation needs to be verified. This point has not yet, to the best of our knowledge, been addressed in the literature on CVs. Our next result below provides regularity conditions for the existence of a solution when using $\mathcal{L}_{\mathrm{SL}}$, the Stein operator used in our experiments. Denote the Sobolev space $W^{k,p}(\Pi)$ of functions whose weak derivatives of order k are in $L^p(\Pi)$ and the Sobolev space $W^{k,p}_{\mathrm{loc}}$ of functions whose p-th power weak derivatives of order k are locally integrable; these are formally defined in Appendix 7.1. For a vector-valued function $h : \mathbb{R}^d \to \mathbb{R}^p$ we let $\|h\|_{L^p(\Pi)} := (\sum_{i=1}^d \|h_i\|^2_{L^p(\Pi)})^{1/2}$.

Proposition 2 *Consider the vector space $\mathcal{U} = W^{2,2}(\Pi) \cap W^{1,4}(\Pi)$ equipped with norm $\|u\|_{\mathcal{U}} := \max(\|u\|_{W^{1,4}(\Pi)}, \|u\|_{W^{2,2}(\Pi)})$. Then $\mathcal{L}_{\mathrm{SL}} : \mathcal{U} \to L^2(\Pi)$ is a bounded linear operator with $\|\mathcal{L}_{\mathrm{SL}}\|_{\mathcal{U}\to L^2(\Pi)} \le 2(\|\nabla \log \pi\|^2_{L^4(\Pi)} + 1)^{\frac{1}{2}}$.*

Furthermore, suppose that

(i) $\int \|x\|_2^K \mathrm{d}\Pi(x) < \infty$ *for some* $K > 8$,
(ii) $(\nabla \log \pi)(x) \cdot (x/\|x\|_2) \le -r\|x\|_2^\alpha$ *for some* $\alpha > -1, r > 0$, *and all* $\|x\|_2 > M$ *for some* $M > 0$,
(iii) $|f(x)| \le C_1 + C_2\|x\|_2^\beta$ *for some* $C_1, C_2 \ge 1$ *and* $\beta < K/4 - 2$.

Then, $\exists u \in \mathcal{U}$ that solves the Stein equation $\mathcal{L}_{\mathrm{SL}}u = f - \Pi[f]$.

The fact that the space $\mathcal{U}$ in Theorem 2 is separable ensures that suitable approximating sets $\mathcal{V}_p$ can be constructed. For example, if $\{u_i\}_{i=1}^\infty$ is a spanning set for $\mathcal{U}$ then we may set $\mathcal{V}_p = \mathrm{span}(u_1, \ldots, u_p)$, in which case $\cup_{p\in\mathbb{N}}\mathcal{V}_p$ is dense in $\mathcal{U}$ so the result of Theorem 2 holds.

Notice that a solution to the Stein equation will not be unique, since one can introduce an additive constant. This motivates, in practice, the use of an additional regularizer $\Omega(\theta)$ to ensure uniqueness of the minimum of $\theta \mapsto J(v_\theta)$.

In [50] we also recall a standard convergence result for SGD in settings where the objective is convex, focusing on the case where $\mathcal{G}$ is a finite dimensional linear space. This result is thus applicable to polynomials and kernels, but not NN-based CVs.

Table 1 *Mean absolute error (based on 20 repetitions) for polynomial-based CV, kernel-based CV and an ensemble of these, for the Genz benchmark* [18]. We took $n = 1000$, $m = 500$ and $d = 1$. The training time presented is for 25 epochs, averaged over repetitions for all integrands

Integrand f	MC	Poly. CV	Ker. CV	Poly.+Ker. CV
Continuous	2.77e–03	3.21e–03	3.28e–04	1.85e–04
Corner peak	5.76e–03	1.07e–03	9.27e–06	6.05e–06
Discontinuous	2.04e–02	1.32e–02	3.91e–03	2.65e–03
Gaussian peak	1.47e–03	1.40e–03	1.24e–05	1.05e–05
Oscillatory	4.17e–03	1.06e–03	4.63e–06	3.90e–06
Product peak	1.37e–03	1.32e–03	2.12e–05	2.52e–06
Time (s)	7.10e–02	4.30e+00	2.60e+00	5.70e+00

5 Empirical Assessment

Here we assess our method on both synthetic problems and on problems arising in a Bayesian statistical context. Our aim is twofold; (i) to assess whether our learning procedure provides a speed-up compared to existing approaches, and (ii) to gain insight into which class of CV may be most appropriate for a given context. The Stein operator $\mathcal{L}_{\mathrm{SL}}$ was used for all experiments. For the polynomial and kernel CVs, the regularizer $\Omega(\theta) = \|\theta\|_2^2$ was used, while for NN CVs the regularizer $\Omega(\theta) = \sum_{i=1}^{m} g_\theta(x_i)^2$ was used, following [54]. The regularization strength parameter λ was tuned by cross-validation. For some datasets, we employed two ensemble CVs: a sum of kernel and a polynomial (i.e., kernel + polynomial); and a sum including two kernels with different hyperparameters and a polynomial (i.e., multiple kernels + polynomial). Implementation details and further experiments are provided in [50].

Genz Test Functions:
The Genz functions are a standard benchmark used to evaluate a numerical integration method [18]. These functions f exhibit discontinuities and sharp peaks, but nevertheless they can be exactly integrated. The purpose of this first experiment is simply to assess whether *any* variance reduction can be achieved using our general framework in challenging and pathological situations.[2] Results are shown in Table 1 for polynomial-based and kernel-based CVs, as well as an ensemble of both. The CVs are trained using SGD on the least-squares objective functional with batch size $b = 8$ for 25 epochs. For each f, the mean absolute error (MAE) of polynomial CVs is always the largest while the linear combination of kernel and polynomial consistently performs the best. This is likely due to the increased flexibility of the CV. In all cases a substantial reduction in MAE was achieved, compared to MC. Full details and an extensive range of additional experiments are provided in [50].

[2] We emphasize that MC can be evaluated at negligible cost and we are not advocating that our methods should be preferred for this task.

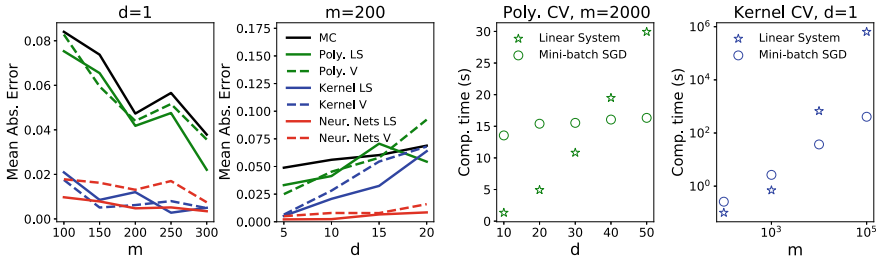


Fig. 2 *Integrating Gaussian Processes. Left and centre-left:* The mean absolute error (based on 20 repetitions) of the CV estimators as a function of the training set size m and dimension d. *Centre-right and right:* Compute times for polynomial and kernel CVs as a function of m and d

Integrating Gaussian Processes:
To automatically generate test problems, we modeled f as a Gaussian process (GP) and sampled $(\Pi[f], f(x_1), \ldots, f(x_n))$ from its Gaussian marginal; here the GP was centered and a squared-exponential covariance function was used, and the distribution Π was taken to be an L-component Gaussian mixture model. In this way infinitely many problem instances can be generated, of a similar nature to those arising in computer experiments [27] and Bayesian numerical methods [10, 41]. We compared CVs based on polynomials, kernels, and NNs (three-layer ResNet with ReLU activation with 50 neurons per layer).

Results are presented in Fig. 2, with implementational details in [50]. The left-most panel presents the performance of each CV for minimizing either $\tilde{J}_m^{\mathrm{V}}$ or $\tilde{J}_m^{\mathrm{LS}}$ in $d = 1$. Polynomials are not flexible enough for such complex integrands, but kernels and NNs can achieve substantial reduction in error. However, we found that the "effective" time requires to implement a NN, including initialization of SGD and selecting an appropriate learning rate, meant that NN were not time-competitive with the other methods considered. The center-left panel studies the impact of d on the performance of each method. The performance of polynomial and kernel CVs degrades rapidly with d, but this is not the case for NNs. In both panels, $\tilde{J}_m^{\mathrm{LS}}$ leads to improved results compared to $\tilde{J}_m^{\mathrm{V}}$. The centre-right and right panels report computational times of linear system and mini-batch SGD as d and m grows. These two panels verify that mini-batch SGD has linear time complexity as n or d is increased, whist exact solution of linear systems leads to exponential computational costs for polynomial and kernel CVs.

Parameter Inference for Ordinary Differential Equations:
Here we consider the problem of inference for parameters $\alpha, \beta, \gamma, \delta$ of the Lotka-Volterra equations $\dot{x} = \alpha x - \beta xy$, $\dot{y} = \delta xy - \gamma y$, a popular ecological model for competing populations. Our experimental set up is identical to that used in [48]. Our task is to compute posterior means of these dynamic parameters based on datasets of size n arising as a subsample from Metropolis-adjusted Langevin algorithm output; the full MCMC output provided the ground truth. Half of the sample was used to

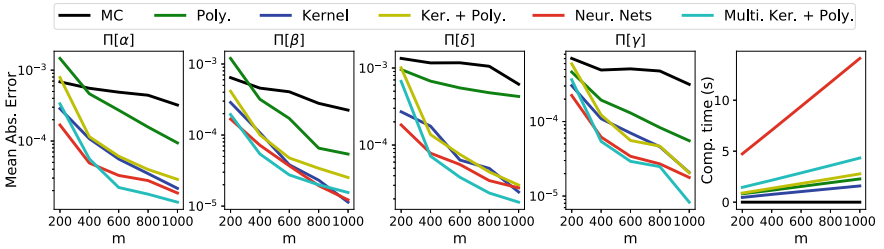


Fig. 3 *Parameter Inference for Ordinary Differential Equations.* Each panel except the rightmost presents the mean absolute error (based on 20 repetitions) for approximation of posterior expectations of model parameters using MCMC output. The rightmost panel presents the computing time of training these CVs. Here "MC" represents the benchmark where no CV is used

train CVs ($m = \frac{n}{2}$) and a batch size of $b = 8$ was used over 25 epochs in SGD based on the least-squares objective functional.

Figure 3 displays the performance of different CVs under sizes of training dataset. In each case the standard MC estimate is outperformed, with ensemble of multiple kernels with a polynomial or the NN performing uniformly best. Due to the computational cost of training NNs as shown in the rightmost panel, we found the ensemble to be preferable. The ensemble also leads to a convex objective which is easier to minimize.

High-dimensional Bayesian Logistic Regression

In this final example, we consider Bayesian logistic regression. We experimented on two different datasets: the Sonar data and the Madelon data. The Sonar dataset has dimension $d = 61$, which is lower than the $d = 500$ of the Madelon dataset. Results were similar for both experiments, and the Sonar data is therefore relegated to [50].

The Madelon data is an artificial dataset, which was part of the NIPS/NeurIPS 2003 feature selection challenge. This is a two-class classification problem with 500 continuous input variables. We denote by β the weight vector that includes all parameters to infer in the Bayesian logistic regression. MCMC was used to sample from the posterior of β. Our task is to approximate the posterior probability that an unlabeled data point z corresponds to label 1, rather than 0, based on a subset of size m from the MCMC output. Thus $f(\beta) = (1 + \exp(-z^\top \beta))^{-1}$. The entire chain was used to establish "ground truth" for the value of this integral.

In these experiments, J_m^{LS} was used with $m = n$ and batch sizes of $b = 8$ over 25 epochs of SGD. Figure 4 compares the performance of different CV methods. The two ensemble CVs and the NNs perform significantly better than other CVs. When $m < 1000$, the NNs and the CV with multiple kernels and a polynomial have similar performance, better than others. When $m \geq 1000$, the ensemble CV surpasses NNs. One possible explanation is that for all values of m we used the same multilayer perceptron with 6 layers and 20 nodes in each of them. Therefore, the NNs size (capacity) remains the same while the training data size m increases. Further growing the depth of NN could lead to an improved performance. Furthermore, the

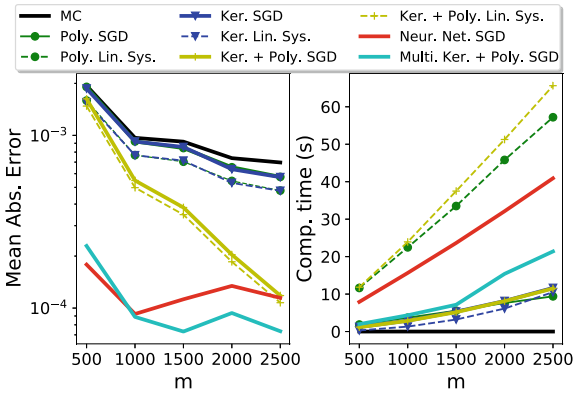


Fig. 4 *Madelon Dataset*. The mean absolute error (left) and compute times (right), as a function of the size m of the training set; based on 20 repetitions

results for polynomials and kernels demonstrate that our general framework based on SGD can achieve comparable MAE with exactly solving the linear systems, but with a fraction of the associated computational overhead. The compute time of NN in Fig. 4 does not capture the time required to manually calibrate SGD, so that the "effective" compute time is much higher than reported.

6 Conclusion

This paper outlined a general framework for developing CVs using Stein operators and SGD. It was demonstrated that (i) the proposed training scheme leads to speed-ups compared to existing CV methods; (ii) novel CV methods (e.g., ensemble methods) can be easily developed; (iii) theoretical analysis can be performed in quite a general setting that simultaneously encompasses multiple CV methods. Further research could explore the use of other Stein classes and operators. In terms of Stein classes, one could consider the use of wavelets, which are known for their good performance for multi-scale function approximation, or other NN architectures which could provide further gains in high dimensions. Stein operators are not unique and one could explore parameterized operators [29] and include these parameters in the optimization scheme. Finally, one could construct novel CVs on other spaces, such as general smooth manifolds or countable spaces [6].

7 Appendix

This appendix contains proofs for the results in the main text.

7.1 Some Elements from Functional Analysis

Let X and Y be two normed real vector spaces. A function $f : X \to Y$ is called *Lipschitz* continuous if there exists a constant L such that, $\forall x, x' \in X$: $\|f(x) - f(x')\|_Y \leq L\|x - x'\|_X$. The smallest such $L \geq 0$ is called the *Lipschitz constant* of f. The norm of a bounded linear operator $\mathcal{L} : X \to Y$ is given by: $\|\mathcal{L}\|_{X\to Y} := \inf\{c \geq 0 : \|\mathcal{L}x\| \leq c\|y\| \ \forall x \in X\}$. For $1 \leq p < \infty$ we denote

$$L^p(\Pi) := \left\{ f : \mathbb{R}^d \to \mathbb{R} \text{ measurable} \,\middle|\, \|f\|_{L^p(\Pi)} := \left(\int_{\mathbb{R}^d} |f(x)|^p \Pi(\mathrm{d}x) \right)^{\frac{1}{p}} < \infty \right\}$$

$$L^p_{\mathrm{loc}} := \left\{ f : \mathbb{R}^d \to \mathbb{R} \text{ measurable} \,\middle|\, \left(\int_K |f(x)|^p \mathrm{d}x \right)^{\frac{1}{p}} < \infty, \ \forall \text{compact } K \subset \mathbb{R}^d \right\}.$$

As usual, $L^p(\Pi)$ can be interpreted as a normed space via identification of functions that agree Π-almost everywhere on $\mathbb{R}^d$. Using this definition, we can now also define weighted Sobolev spaces of integer smoothness:

$$W^{k,p}(\Pi) := \left\{ f \in L^p(\Pi) \,\middle|\, D^\alpha f \in L^p(\Pi) \ \forall |\alpha| \leq k \right\}$$

$$W^{k,p}_{\mathrm{loc}} := \left\{ f \in L^p_{\mathrm{loc}} \,\middle|\, D^\alpha f \in L^p_{\mathrm{loc}} \ \forall |\alpha| \leq k \right\}$$

In this definition, $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ is a multi-index and D^α denotes the weak derivative of order α, *i.e.* $D^\alpha f := \partial^{|\alpha|} f / \partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}$. Recall that $W^{k,p}(\Pi)$ can be interpreted as a normed space with norm

$$\|u\|_{W^{k,p}(\Pi)} := \left(\sum_{i=0}^{k} \sum_{\alpha : |\alpha| = i} \int |D^\alpha u(x)|^p \mathrm{d}\Pi(x) \right)^{\frac{1}{p}},$$

again via identification of functions whose derivatives up to order $|\alpha| \leq k$ agree Π-almost everywhere on $\mathbb{R}^d$.

7.2 Proof of Proposition 1

Proof Let $u \in \mathcal{U}$ solve the Stein equation $\mathcal{L}u = f - \Pi[f]$. Since $\mathcal{L}$ is a bounded linear operator between normed spaces,

$$J(v) = \|f - \Pi[f] - \mathcal{L}v\|^2_{L^2(\Pi)} = \|\mathcal{L}u - \mathcal{L}v\|^2_{L^2(\Pi)} \leq \|\mathcal{L}\|^2_{\mathcal{U}\to L^2(\Pi)} \|u - v\|^2_{\mathcal{U}}$$

where $\|\mathcal{L}\|_{\mathcal{U}\to\mathcal{L}^2(\Pi)} < \infty$. Fix $\epsilon > 0$. Since $u \in \mathcal{U}$ and $\cup_{p\in\mathbb{N}} \mathcal{V}_p$ is dense in $\mathcal{U}$, there exists $v \in \cup_{p\in\mathbb{N}} \mathcal{V}_p$ such that $\|u - v\|_{\mathcal{U}} < \epsilon$. In particular, there exists $q \in \mathbb{N}$ such

that $v \in \mathcal{V}_q$. Moreover, since $\mathcal{V}_q \subseteq \mathcal{V}_p$ for all $q \leq p$, the function $p \mapsto \inf_{v \in \mathcal{V}_p} J(v)$ is non-increasing. Thus

$$0 \leq \lim_{p \to \infty} \inf_{v \in \mathcal{V}_p} J(v) \leq \inf_{v \in \mathcal{V}_q} J(v) \leq \|\mathcal{L}\|^2_{\mathcal{U} \to L^2(\Pi)} \inf_{v \in \mathcal{V}_q} \|u - v\|^2_{\mathcal{U}} \leq \|\mathcal{L}\|^2_{\mathcal{U} \to L^2(\Pi)} \epsilon^2.$$

Since $\epsilon > 0$ was arbitrary, the right hand side can be made arbitrarily small.

7.3 Proof of Proposition 2

Proof First we will show that $\mathcal{L}_{\mathrm{SL}}$ is a bounded linear operator from $\mathcal{U} = W^{2,2}(\Pi) \cap W^{1,4}(\Pi)$ to $L^2(\Pi)$. To this end:

$$\begin{aligned}
\|\mathcal{L}_{\mathrm{SL}} u - \mathcal{L}_{\mathrm{SL}} v\|^2_{L^2(\Pi)} &= \|\nabla \log \pi \cdot \nabla(u - v) + \nabla \cdot \nabla(u - v)\|^2_{L^2(\Pi)} && (3)\\
&\leq 2\left[\|\nabla \log \pi \cdot \nabla(u - v)\|^2_{L^2(\Pi)} + \|\nabla \cdot \nabla(u - v)\|^2_{L^2(\Pi)}\right] && (4)\\
&\leq 2\left[\|\nabla \log \pi\|^2_{L^4(\Pi)} \|\nabla(u - v)\|^2_{L^4(\Pi)} + \|u - v\|^2_{W^{2,2}(\Pi)}\right] && (5)\\
&\leq 2\left(\|\nabla \log \pi\|^2_{L^4(\Pi)} + 1\right)\left(\|u - v\|^2_{W^{1,4}(\Pi)} + \|u - v\|^2_{W^{2,2}(\Pi)}\right), && (6)\\
&\leq 4\left(\|\nabla \log \pi\|^2_{L^4(\Pi)} + 1\right) \max\left(\|u - v\|_{W^{1,4}(\Pi)}, \|u - v\|_{W^{2,2}(\Pi)}\right)^2 && (7)
\end{aligned}$$

Equation (3) follows by definition of the Stein operator, Equation (4) follows from the fact that $(a + b)^2 \leq 2(a^2 + b^2)$. Equation (5) follows from the vector-valued Hölder inequality together with the definition of $\|\cdot\|_{W^{2,2}(\Pi)}$. Equation (6) follows from the definition of $\|\cdot\|_{W^{1,4}(\Pi)}$. Thus $\mathcal{L}_{\mathrm{SL}}$ is a bounded linear operator as claimed, and moreover $\|\mathcal{L}_{\mathrm{SL}}\|_{\mathcal{U} \to L^2(\Pi)} \leq 2(\|\nabla \log \pi\|^2_{L^4(\Pi)} + 1)^{\frac{1}{2}}$.

The second task is to establish that there exists a solution to the Stein equation $\mathcal{L}_{\mathrm{SL}} u = f - \Pi[f]$. For this we leverage [44, Theorem 1] which states that, if conditions (ii), (iii) hold, there exists a solution u to the Stein equation which is continuous and belongs to $W^{2,q}_{\mathrm{loc}}$ for all $q > 1$. Moreover, $\forall m > \beta + 2$ there exists C_m such that $|u(x)| + |\nabla u(x)| \leq C_m(1 + |x|^m)$ for all $x \in \mathbb{R}^d$. By assumption (i) it follows that $u \in W^{1,4}(\Pi)$. Moreover, since π was assumed to be smooth (recall, this was assumed at the outset in Sect. 1), standard regularity results imply that u is smooth and so, is a classical solution. We can therefore write

$$|\Delta u(x)| \leq |f(x)| + |\Pi(f)| + |\nabla \log \pi(x) \cdot \nabla u(x)|, \quad x \in \mathbb{R}^d,$$

so that $\|\Delta u\|_{L^2(\Pi)} \leq 2\|f\|_{L^2(\Pi)} + \|\nabla \log \pi\|_{L^4(\Pi)} \|u\|_{W^{1,4}(\Pi)} < \infty$. It follows that $u \in W^{2,2}(\Pi) \cap W^{1,4}(\Pi)$, as claimed.

Acknowledgements The authors would like to thank Charline Le Lan for helpful discussions, and Wilson Chen, Marina Riabiz and Leah South for sharing MCMC samples from the model

of atmospheric pollutants, the predator-prey model and the logistic regression model respectively. CJO, ABD, FXB were also supported by the Lloyd's Register Foundation Programme on Data-Centric Engineering and the Alan Turing Institute under the EPSRC grant [EP/N510129/1]. FXB was supported by an Amazon Research Award on "Transfer Learning for Numerical Integration in Expensive Machine Learning Systems".

References

1. Andradóttir, S., Heyman, D.P., Ott, T.J.: Variance reduction through smoothing and control variates for Markov chain simulations. ACM Trans. Model. Comput. Simul. **3**(3), 167–189 (1993)
2. Assaraf, R., Caffarel, M.: Zero-variance principle for Monte Carlo algorithms. Phys. Rev. Lett. **83**(23), 4682 (1999)
3. Baker, J., Fearnhead, P., Fox, E.B., Nemeth, C.: Control variates for stochastic gradient MCMC. Stat. Comput. **29**, 599–615 (2019)
4. Barbour, A.D.: Stein's method and Poisson process convergence. J. Appl. Probab. **25**, 175–184 (1988)
5. Barp, A., Briol, F.X., Duncan, A.B., Girolami, M., Mackey, L.: Minimum Stein discrepancy estimators. In: Neural Information Processing Systems, pp. 12964–12976 (2019)
6. Barp, A., Oates, C.J., Porcu, E., Girolami, M.: A Riemannian-Stein Kernel Method. arXiv:1810.04946 (2018)
7. Belomestny, D., Iosipoi, L., Moulines, E., Naumov, A., Samsonov, S.: Variance reduction for Markov chains with application to MCMC. Stat. Comput. **30**, 973–997 (2020)
8. Belomestny, D., Iosipoi, L., Zhivotovskiy, N.: Variance reduction via empirical variance minimization: convergence and complexity. Doklady Math. **98**, 494–497 (2018)
9. Belomestny, D., Moulines, E., Shagadatov, N., Urusov, M.: Variance Reduction for MCMC Methods Via Martingale Representations (2019). arXiv:1903.0737
10. Briol, F.X., Oates, C.J., Girolami, M., Osborne, M.A., Sejdinovic, D.: Probabilistic integration: a role in statistical computation? (with discussion). Stat. Sci. **34**(1), 1–22 (2019)
11. Brosse, N., Durmus, A., Meyn, S., Moulines, E.: Diffusion approximations and control variates for MCMC (2018). arXiv:1808.01665
12. Chen, L.H.Y., Goldstein, L., Shao, Q.M.: Normal Approximation by Stein's Method. Springer, Berlin (2010)
13. Chen, W.Y., Barp, A., Briol, F.X., Gorham, J., Girolami, M., Mackey, L., Oates, C.J.: Stein point Markov chain Monte Carlo. In: International Conference on Machine Learning, PMLR 97, pp. 1011–1021 (2019)
14. Chen, W.Y., Mackey, L., Gorham, J., Briol, F.X., Oates, C.J.: Stein points. In: Proceedings of the International Conference on Machine Learning, PMLR 80:843–852 (2018)
15. Chwialkowski, K., Strathmann, H., Gretton, A.: A kernel test of goodness of fit. Int. Conf. Mach. Learn. **48**, 2606–2615 (2016)
16. Dellaportas, P., Kontoyiannis, I.: Control variates for estimation based on reversible Markov chain Monte Carlo samplers. J. R. Stat. Soc. Ser. B: Stat. Methodol. **74**(1), 133–161 (2012)
17. Friel, N., Mira, A., Oates, C.J.: Exploiting multi-core architectures for reduced-variance estimation with intractable likelihoods. Bayesian Anal. **11**(1), 215–245 (2014)
18. Genz, A.: Testing multidimensional integration routines. In: Proceedings of the International Conference on Tools, Methods and Languages for Scientific and Engineering Computation, pp. 81–94 (1984)
19. Gorham, J., Duncan, A., Mackey, L., Vollmer, S.: Measuring sample quality with diffusions. Ann. Appl. Probab. **29**(5), 2884–2928 (2019)
20. Gorham, J., Mackey, L.: Measuring sample quality with Stein's method. In: Advances in Neural Information Processing Systems, pp. 226–234 (2015)

21. Gorham, J., Mackey, L.: Measuring sample quality with kernels. In: Proceedings of the International Conference on Machine Learning, pp. 1292–1301 (2017)
22. Grathwohl, W., Choi, D., Wu, Y., Roeder, G., Duvenaud, D.: Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In: International Conference on Learning Representations (2018)
23. Greensmith, E., Bartlett, P.L., Baxter, J.: Variance reduction techniques for gradient estimates in reinforcement learning. J. Mach. Learn. Res. **5**, 1471–1530 (2004)
24. Hammer, H., Tjelmeland, H.: Control variates for the Metropolis-Hastings algorithm. Scand. J. Stat. **35**(3), 400–414 (2008)
25. Henderson, S.G., Glynn, P.W.: Approximating martingales for variance reduction in Markov process simulation. Math. Oper. Res. **27**(2), 253–271 (2002)
26. Hickernell, F.J., Lemieux, C., Owen, A.B.: Control variates for quasi-Monte Carlo. Stat. Sci. **20**(1), 1–31 (2005)
27. Kennedy, M.C., Hagan, A.O.: Bayesian calibration of computer models. J. R. Stat. Soc. Ser. B: Stat. Methodol. **63**(3), 425–464 (2001)
28. Leluc, R., Portier, F., Segers, J.: Control variate selection for Monte Carlo integration (2019). arXiv:1906.10920
29. Ley, C., Swan, Y.: Parametric Stein operators and variance bounds. Braz. J. Probab. Stat. **30**(2) (2016)
30. Liu, H., Feng, Y., Mao, Y., Zhou, D., Peng, J., Liu, Q.: Action-dependent control variates for policy optimization via Stein's identity. In: International Conference on Learning Representation (2018)
31. Liu, Q., Lee, J.D.: Black-box importance sampling. In: Proceedings of the International Conference on Artificial Intelligence and Statistics, pp. 952–961 (2017)
32. Liu, Q., Lee, J.D., Jordan, M.I.: A kernelized Stein discrepancy for goodness-of-fit tests and model evaluation. In: International Conference on Machine Learning, pp. 276–284 (2016)
33. Liu, Q., Wang, D.: Stein variational gradient descent: a general purpose Bayesian inference algorithm. In: Advances in Neural Information Processing Systems (2016)
34. Liu, S., Kanamori, T., Jitkrittum, W., Chen, Y.: Fisher efficient inference of intractable models. In: Neural Information Processing Systems, pp. 8793–8803 (2019)
35. Mira, A., Solgi, R., Imparato, D.: Zero variance Markov chain Monte Carlo for Bayesian estimators. Stat. Comput. **23**(5), 653–662 (2013)
36. Müller, T., Rousselle, F., Keller, A., Novák, J.: Neural control variates. ACM Trans. Graph. **39**(6), 243:1–243:19 (2020). https://doi.org/10.1145/3414685.3417804
37. Newton, N.J.: Variance reduction for simulated diffusions. SIAM J. Appl. Math. **54**(6), 1780–1805 (1994)
38. Oates, C.J., Cockayne, J., Briol, F.X., Girolami, M.: Convergence rates for a class of estimators based on Stein's identity. Bernoulli **25**(2), 1141–1159 (2019)
39. Oates, C.J., Girolami, M., Chopin, N.: Control functionals for Monte Carlo integration. J. R. Stat. Soc. B: Stat. Methodol. **79**(3), 695–718 (2017)
40. Oates, C.J., Papamarkou, T., Girolami, M.: The controlled thermodynamic integral for Bayesian model comparison. J. Am. Stat. Assoc. (2016)
41. O'Hagan, A.: Bayes-Hermite quadrature. J. Stat. Plan. Inference **29**, 245–260 (1991)
42. Paisley, J., Blei, D., Jordan, M.: Variational Bayesian inference with stochastic search. In: International Conference on Machine Learning (2012)
43. Papamarkou, T., Mira, A., Girolami, M.: Zero variance differential geometric Markov chain Monte Carlo algorithms. Bayesian Anal. **9**(1), 97–128 (2014)
44. Pardoux, E., Vertennikov, A.Y.: On the Poisson equation and diffusion approximation. I. Ann. Probab. **29**(3), 1061–1085 (2001)
45. Portier, F., Segers, J.: Monte Carlo integration with a growing number of control variates. J. Appl. Probab. **56**(4), 1168–1186 (2019)
46. Ranganath, R., Altosaar, J., Tran, D., Blei, D.M.: Operator variational inference. In: Advances in Neural Information Processing Systems, pp. 496–504 (2016)

47. Ranganath, R., Gerrish, S., Blei, D.M.: Black box variational inference. In: Artificial Intelligence and Statistics, pp. 814–822 (2014)
48. Riabiz, M., Chen, W., Cockayne, J., Swietach, P., Niederer, S.A., Mackey, L., Oates, C.J.: Optimal thinning of MCMC output (2020). arXiv:2005.03952
49. Ross, N.: Fundamentals of Stein's method. Probab. Surv. **8**, 210–293 (2011)
50. Si, S., Oates, C.J., Duncan, A.B., Carin, L., Briol, F.X.: Scalable Control Variates for Monte Carlo Methods via Stochastic Optimization (2020). arXiv:2006.07487
51. South, L.F., Karvonen, T., Nemeth, C., Girolami, M., Oates, C.J.: Semi-exact control functionals from Sard's method (2020). arXiv:2002.00033
52. South, L.F., Oates, C.J., Mira, A., Drovandi, C.: Regularised zero-variance control variates for high-dimensional variance reduction (2019). arXiv:1811.05073
53. Stein, C.: A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In: Proceedings of 6th Berkeley Symposium on Mathematical Statistics and Probability, pp. 583–602. University of California Press (1972)
54. Wan, R., Zhong, M., Xiong, H., Zhu, Z.: Neural control variates for Monte Carlo variance reduction. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 533–547 (2019)
55. Wang, C., Chen, X., Smola, A., Xing, E.P.: Variance reduction for stochastic gradient optimization. In: Advances in Neural Information Processing Systems, pp. 181–189 (2013)
56. Yang, J., Liu, Q., Rao, V., Neville, J.: Goodness-of-fit testing for discrete distributions via Stein discrepancy. In: International Conference on Machine Learning, pp. 5561–5570 (2018)

Simulation of Conditional Expectations Under Fast Mean-Reverting Stochastic Volatility Models

Andrei S. Cozma and Christoph Reisinger

Abstract We study the simulation of a large system of stochastic processes subject to a common driving noise and fast mean-reverting stochastic volatilities. This model may be used to describe the firm values of a large pool of financial entities. We then seek an efficient estimator for the probability of a default, indicated by a firm value below a certain threshold, conditional on common factors. We consider approximations where coefficients containing the fast volatility are replaced by certain ergodic averages (a type of law of large numbers), and study a correction term (of central limit theorem-type). The accuracy of these approximations is assessed by numerical simulation of pathwise losses and the estimation of payoff functions as they appear in basket credit derivatives.

Keywords Particle systems · Common noise · Multiple time scales · Ergodicity · Stochastic filtering · Basket credit derivatives

1 Introduction and Preliminaries

Consider a complete filtered probability space that is the product of two independent probability spaces,

$$(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P}) = (\Omega^{x,y} \times \Omega^{\dagger}, \mathcal{F}^{x,y} \otimes \mathcal{F}^{\dagger}, \{\mathcal{F}_t^{x,y} \otimes \mathcal{F}_t^{\dagger}\}_{t\geq 0}, \mathbb{P}^{x,y} \times \mathbb{P}^{\dagger}),$$

such that $(\Omega^{x,y}, \mathcal{F}^{x,y}, \{\mathcal{F}_t^{x,y}\}_{t\geq 0}, \mathbb{P}^{x,y})$ supports a two-dimensional standard Brownian motion (W^x, W^y) adapted to $\{\mathcal{F}_t^{x,y}\}_{t\geq 0}$ and with correlation $-1 < \rho_{xy} < 1$, and $(\Omega^{\dagger}, \mathcal{F}^{\dagger}, \{\mathcal{F}_t^{\dagger}\}_{t\geq 0}, \mathbb{P}^{\dagger})$ supports an infinite i.i.d. sequence of two-dimensional uncorrelated standard Brownian motions $(W^{x,i}, W^{y,i})_{i\geq 1}$ adapted to $\{\mathcal{F}_t^{\dagger}\}_{t\geq 0}$.

A. S. Cozma · C. Reisinger (✉)
Mathematical Institute, University of Oxford, Woodstock Road,
OX2 6GG Oxford, United Kingdom
e-mail: christoph.reisinger@maths.ox.ac.uk

A. S. Cozma
e-mail: andrei.s.cozma@gmail.com

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_11

For positive $N_f \in \mathbb{Z}$, we study an $N_f \times 2$-dimensional system of SDEs of the form

$$\begin{aligned} \mathrm{d}X_t^i &= \mu(V_t^i)\,\mathrm{d}t + \sigma(V_t^i)\Big(\rho_x\,\mathrm{d}W_t^x + \sqrt{1-\rho_x^2}\,\mathrm{d}W_t^{x,i}\Big), \\ \mathrm{d}V_t^i &= -\frac{k}{\epsilon}V_t^i\,\mathrm{d}t + \frac{g(V_t^i)}{\sqrt{\epsilon}}\Big(\rho_y\,\mathrm{d}W_t^y + \sqrt{1-\rho_y^2}\,\mathrm{d}W_t^{y,i}\Big), \end{aligned} \tag{1}$$

with $\rho_x, \rho_y \in (-1, 1)$, $\epsilon, \kappa > 0$ all constant; $\mu : \mathbb{R} \to \mathbb{R}$ and $\sigma, g : \mathbb{R} \to \mathbb{R}_+$ given functions; $((X_0^i, V_0^i))_{i\geq 1}$ are an exchangeable infinite sequence of two-dimensional random variables that are measurable with respect to $\mathcal{F}_0 = \mathcal{F}_0^{x,y} \otimes \mathcal{F}_0^{\dagger}$.

We will consider the marginal distribution of any X_t^i, conditional on $\mathcal{F}_t^{x,y}$, which is the reason for writing the Brownian driver in the decomposed way above. Specifically, we study the setting of small ϵ, a characteristic, dimensionless reversion time of V to its mean. The mean is chosen 0 here without loss of generality, but the general case is obtained by adding the constant mean to Y and re-defining σ and μ.

The process X is thought to describe the log-asset prices of a large portfolio of financial entities and V their instantaneous stochastic volatilities. The event of X^i being below a certain threshold, or barrier, B models the default of that entity. Therefore, estimating marginal distributions of X^i conditional on the market factors is important for the valuation and risk management of basket credit derivatives.

A simplified version of X in (1) with constant σ has been considered in [4], where an SPDE for the empirical measure in the large pool limit is derived and used to compute tranche spreads of collateralised debt obligations, extended to jump-diffusions in [3]. The multilevel estimation of conditional expectations using the SDE system is analysed in [2], and a multilevel scheme for the SPDE in [9].

The large pool limit under stochastic volatilities is studied in [10]. Computationally, this presents extra difficulties partly because of the extra dimension of the conditional expectations, but also because empirical data demonstrate a fast timescale in the volatility component (see [6–8]), which makes accurate simulation substantially more time consuming. Motivated by the earlier work above on ergodic limits in the context of derivative pricing (and hence parabolic PDEs), [12] derive convergence in distribution of the conditional law of X as $\epsilon \to 0$, leading to an SDE with coefficients averaged over the ergodic measure of the fast volatility process.

In this paper, we first present in Sect. 2 the simulation schemes used, including the standard Euler-Maruyama scheme and an improved scheme which exploits exact integration of the fast process. We then investigate in Sect. 3 a number of approximations to the X process where the coefficients depending on V are replaced by certain ergodic averages, and give an application to credit derivatives in Sect. 4. Moreover, we compute novel correction terms, heuristically motivated by a central limit theorem-type argument, which are shown to give significantly improved results, across all scenarios considered.

For simplicity, we will restrict ourselves to the case of constant $g = \sqrt{2}\xi$, i.e., an Ornstein-Uhlenbeck (O–U) process V, and $X_0^i = 0$ and $Y_0^i = y_0$ deterministic for all i. In that case, if we introduce a process Z as the (strong) solution to

$$\mathrm{d}Z_t = -\frac{k}{\epsilon} Z_t \,\mathrm{d}t + \frac{\xi\sqrt{2}}{\sqrt{\epsilon}} \rho_y \,\mathrm{d}W_t^y, \qquad Z_0 = 0, \tag{2}$$

then (X, Y) with $Y := V - Z$ satisfies, for $1 \le i \le N_f$,

$$\begin{cases} dX_t^i & = \mu(Y_t^i + Z_t)\,dt + \sigma(Y_t^i + Z_t)\Big(\rho_x dW_t^x + \sqrt{1-\rho_x^2}\, dW_t^{x,i}\Big), \quad X_0^i = 0, \\ X_0^i = 0, & \\ dY_t^i & = -\frac{k}{\epsilon} Y_t^i dt + \frac{\xi\sqrt{2}}{\sqrt{\epsilon}} \sqrt{1-\rho_y^2}\, dW_t^{y,i}, \quad Y_0^i = y_0. \end{cases} \tag{3}$$

Consider now the 2-dimensional empirical measure

$$\nu_{N_f,t} = \frac{1}{N_f} \sum_{i=1}^{N_f} \delta_{X_t^i, Y_t^i}. \tag{4}$$

Using exchangeability, [11] prove the existence of a limit measure

$$\nu_t = \lim_{N_f \to \infty} \nu_{N_f,t}, \tag{5}$$

where the weak limit exists almost surely in $\mathbb{P}$, when $\mu = r - \sigma^2/2$ for constant r and continuous bounded σ. This follows [4] for the one-dimensional case of constant volatility, and [10] for stochastic volatility of Cox–Ingersoll–Ross type.

Moreover, for any Borel set A, we have in the set-ups of [4, 10, 11, 13] that

$$\nu_t(A) = \mathbb{P}\left((X_t^1, Y_t^1) \in A \,|\, \mathcal{F}_t^{x,y}\right), \tag{6}$$

where $(\mathcal{F}_t^{x,y})_{t\ge 0}$ is here taken to be the filtration generated by the market Brownian drivers W^x and W^y. Hence, the limit measure can be regarded as the behaviour of a single firm given the market drivers are known.

We expect these results to hold for general μ above also, but do not provide a proof for this as it is not the focus of this paper.

2 Simulation Schemes for the Fast O–U Process

Here, we first give the standard Euler–Maruyama scheme for the fast O–U processes and then give an alternative discretisation based on the closed-from expression for the O–U processes,

$$
\begin{aligned}
Y_t^i &= y_0 e^{-\frac{k}{\epsilon}t} + \frac{\xi\sqrt{2}}{\sqrt{\epsilon}}\sqrt{1-\rho_y^2}\int_0^t e^{-\frac{k}{\epsilon}(t-s)}\,dW_s^{y,i},\\
Z_t &= \frac{\xi\sqrt{2}}{\sqrt{\epsilon}}\rho_y\int_0^t e^{-\frac{k}{\epsilon}(t-s)}\,dW_s^{y}.
\end{aligned}
\tag{7}
$$

For the Euler–Maruyama scheme, we use a time mesh with timestep $\epsilon\delta t$, for some $\delta t > 0$ independent of ϵ. The discrete-time approximation of (Z_t) is thus generated by

$$
\widehat{Z}_n = \widehat{Z}_{n-1} - k\,\delta t\,\widehat{Z}_{n-1} + \frac{\xi\sqrt{2}}{\sqrt{\epsilon}}\rho_y\left(W_{t_n}^y - W_{t_{n-1}}^y\right), \quad n = 1, 2, \ldots, \qquad \widehat{Z}_0 = 0,
\tag{8}
$$

where $t_n = n\delta t\epsilon$, and similar for Y.

The strong error is of order 1 in δt as the diffusion coefficient is constant and the Euler–Maruyama scheme coincides with the Milstein scheme. By choosing the time step proportionally to ϵ, we found empirically that the error is asymptotically independent of ϵ, but the cost increases proportionally to ϵ^{-1}.

In our second scheme, we use the closed-form expressions of Y^i and Z. From (7)

$$
Y_t^i \sim \mathcal{N}\left(y_0 e^{-\frac{k}{\epsilon}t},\ \frac{\xi^2}{k}(1-\rho_y^2)\left(1-e^{-\frac{2k}{\epsilon}t}\right)\right) \overset{\epsilon\to 0}{\to} \mathcal{N}\left(0,\ \frac{\xi^2}{k}(1-\rho_y^2)\right),
\tag{9}
$$

and

$$
Z_t \sim \mathcal{N}\left(0,\ \frac{\xi^2}{k}\rho_y^2\left(1-e^{-\frac{2k}{\epsilon}t}\right)\right) \overset{\epsilon\to 0}{\to} \mathcal{N}\left(0,\ \frac{\xi^2}{k}\rho_y^2\right).
\tag{10}
$$

Furthermore, the processes are independent across time in the limit $\epsilon \to 0$ since they decorrelate exponentially fast on the time scale ϵ (see [7]).

For a fixed time horizon $T > 0$, consider now a uniform grid $t_n = n\delta t$, $n \in \{0, 1, ..., N\}$, where $T = N\delta t$. The discrete-time approximation processes are thus

$$
\begin{aligned}
y_{t_n}^i &= y_0 e^{-\frac{k}{\epsilon}t_n} + \frac{\xi\sqrt{2}}{\sqrt{\epsilon}}\sqrt{1-\rho_y^2}\sum_{j=1}^N e^{-\frac{k}{\epsilon}(t_n-t_{j-1})}\left(W_{t_j}^{y,i} - W_{t_{j-1}}^{y,i}\right)\\
&= e^{-\frac{k}{\epsilon}\delta t}\left(y_{t_{n-1}}^i + \frac{\xi\sqrt{2}}{\sqrt{\epsilon}}\sqrt{1-\rho_y^2}\left(W_{t_n}^{y,i} - W_{t_{n-1}}^{y,i}\right)\right), \quad y_0^i = y_0,
\end{aligned}
\tag{11}
$$

and

$$
\begin{aligned}
z_{t_n} &= \frac{\xi\sqrt{2}}{\sqrt{\epsilon}}\rho_y\sum_{j=1}^N e^{-\frac{k}{\epsilon}(t_n-t_{j-1})}\left(W_{t_j}^{y} - W_{t_{j-1}}^{y}\right)\\
&= e^{-\frac{k}{\epsilon}\delta t}\left(z_{t_{n-1}} + \frac{\xi\sqrt{2}}{\sqrt{\epsilon}}\rho_y\left(W_{t_n}^{y} - W_{t_{n-1}}^{y}\right)\right), \quad z_0 = 0.
\end{aligned}
\tag{12}
$$

Finally, the approximated log-asset price processes are

$$x^i_{t_n} = x^i_{t_{n-1}} + \mu(y^i_{t_{n-1}} + z_{t_{n-1}})\delta t + \sigma(y^i_{t_{n-1}} + z_{t_{n-1}})\Big(\rho_x\Big(W^x_{t_n} - W^x_{t_{n-1}}\Big) + \sqrt{1-\rho_x^2}\Big(W^{x,i}_{t_n} - W^{x,i}_{t_{n-1}}\Big)\Big), \quad x^i_0 = 0. \tag{13}$$

We found in experiments that if we discretize the formulae (7) instead of the SDEs, this yields a lower time-discretization error. We will therefore use the schemes (11) to (13) for the numerical tests in the subsequent sections.

3 Pathwise Conditional CDF

In this section, we give approximations to the loss function $L_T = \mathbb{P}\Big(X^1_T \le B \,|\, \mathcal{F}^{x,y}_T\Big)$, i.e. the CDF of X^1_T conditional on the market factors W^x and W^y, using ergodic averages of coefficients and a correction term from a central limit theorem.

3.1 Conditional CDF and Monte Carlo Estimators

Let $B \in \mathbb{R}$ and consider the loss function at time T for a default level B,

$$L_{N_f,T} = \frac{1}{N_f}\sum_{i=1}^{N_f} \mathbb{1}_{X^i_T \le B}, \tag{14}$$

i.e., the proportion of companies that are in default at time T. Since $\left\{\mathbb{1}_{X^i_T \le B} : 1 \le i \le N_f\right\}$ are conditionally (on $\mathcal{F}^{x,y}_T$) independent and identically distributed random variables, Birkhoff's Ergodic Theorem (see [16, Sect. V.3]) implies that the limiting loss function can be regarded as the marginal CDF, i.e.,

$$L_T = \lim_{N_f\to\infty} L_{N_f,T} = \mathbb{P}\Big(X^1_T \le B \,|\, \mathcal{F}^{x,y}_T\Big). \tag{15}$$

We use a conditional Monte Carlo technique to estimate the marginal CDF. Denote by $\mathcal{F}^{x,y,y_1}$ the filtration generated by the Brownian motions W^x, W^y, $W^{y,1}$. Then

$$\begin{aligned}\mathbb{P}\Big(X^1_T \le B \,|\, \mathcal{F}^{x,y}_T\Big) &= \mathbb{E}\Big[\mathbb{E}\Big[\mathbb{1}_{X^1_T \le B} \,|\, \mathcal{F}^{x,y,y_1}_T\Big] \,\Big|\, \mathcal{F}^{x,y}_T\Big] \\ &= \mathbb{E}\Big[\mathbb{P}\Big(X^1_T \le B \,|\, \mathcal{F}^{x,y,y_1}_T\Big) \,\Big|\, \mathcal{F}^{x,y}_T\Big]. \end{aligned} \tag{16}$$

Conditional on the σ-algebra $\mathcal{F}_T^{x,y,y_1}$, noting $W^{x,1}$ independent of $W^{y,1}$ and W^y,

$$\int_0^T \sigma(Y_t^1 + Z_t)\, dW_t^{x,1} \overset{\text{law}}{=} \sqrt{\int_0^T \sigma^2(Y_t^1 + Z_t)\, dt}\; W_1, \tag{17}$$

where W_1 is a standard normal random variable. Hence, we deduce from (3) that

$$\mathbb{P}\left(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y,y_1}\right) = \Phi\left(\frac{B - \int_0^T \mu(Y_t^1 + Z_t)\, dt - \rho_x \int_0^T \sigma(Y_t^1 + Z_t)\, dW_t^x}{\sqrt{(1-\rho_x^2)\int_0^T \sigma^2(Y_t^1 + Z_t)\, dt}}\right), \tag{18}$$

where Φ is the standard normal CDF. Using the discretizations from (11) and (12),

$$\mathbb{P}\left(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y,y_1}\right) \approx \tag{19}$$

$$\Phi\left(\frac{B - \delta t \sum_{n=0}^{N-1} \mu(y_{t_n}^1 + z_{t_n}) - \rho_x \sum_{n=0}^{N-1} \sigma(y_{t_n}^1 + z_{t_n})\left(W_{t_{n+1}}^x - W_{t_n}^x\right)}{\sqrt{(1-\rho_x^2)\delta t \sum_{n=0}^{N-1} \sigma^2(y_{t_n}^1 + z_{t_n})}}\right). \tag{20}$$

The marginal CDF, i.e., the outer expectation in (16), is estimated by a Monte Carlo average over a sufficiently large number of samples of $W^{y,1}$. As an aside, we can estimate the marginal density function by differentiating (19) with respect to B.

3.2 Ergodic Averages

We will define approximations to the process by averaging SDE coefficients over the ergodic distribution of the O–U process,

$$\langle f \rangle_Y = \int_{-\infty}^{\infty} f(y)\phi_Y(y)\, dy, \tag{21}$$

where ϕ_Y is the centered normal density with variance $\xi^2(1-\rho_y^2)/k$.

Linear Y-average. We first approximate the marginal CDF (in x) by using an ergodic Y^1 average (abbreviated erg_1Y) over its stationary distribution, namely

$$\int_0^T \sigma(Y_t^1 + Z_t)\, dW_t^x \approx \int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y\, dW_t^x, \tag{22}$$

which matches the first conditional (on $\mathcal{F}_T^{x,y}$) moment of the stochastic integral in the limit $\epsilon \to 0$. Hence, we obtain

$$\mathbb{P}\Big(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y}\Big) \approx \Phi\left(\frac{B - \int_0^T \langle \mu(\cdot + Z_t)\rangle_Y \, dt - \rho_x \int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y \, dW_t^x}{\sqrt{(1-\rho_x^2)\int_0^T \langle \sigma^2(\cdot + Z_t)\rangle_Y \, dt}}\right)$$
$$\approx \Phi\left(\frac{B - \delta t \sum_{n=0}^{N-1} \langle \mu(\cdot + z_{t_n})\rangle_Y - \rho_x \sum_{n=0}^{N-1} \langle \sigma(\cdot + z_{t_n})\rangle_Y \left(W_{t_{n+1}}^x - W_{t_n}^x\right)}{\sqrt{(1-\rho_x^2)\delta t \sum_{n=0}^{N-1} \langle \sigma^2(\cdot + z_{t_n})\rangle_Y}}\right). \tag{23}$$

Quadratic Y-average. Alternatively, we will use a quadratic ergodic Y^1 average (abbreviated $\mathrm{erg}_2\mathrm{Y}$), namely

$$\int_0^T \sigma(Y_t^1 + Z_t)\, dW_t^x \approx \int_0^T \langle \sigma^2(\cdot + Z_t)\rangle_Y^{\frac{1}{2}} \, dW_t^x, \tag{24}$$

which matches the first and second unconditional moments of the stochastic integral in the limit $\epsilon \to 0$, to obtain

$$\mathbb{P}\Big(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y}\Big) \approx \Phi\left(\frac{B - \int_0^T \langle \mu(\cdot + Z_t)\rangle_Y \, dt - \rho_x \int_0^T \langle \sigma^2(\cdot + Z_t)\rangle_Y^{\frac{1}{2}} \, dW_t^x}{\sqrt{(1-\rho_x^2)\int_0^T \langle \sigma^2(\cdot + Z_t)\rangle_Y \, dt}}\right)$$
$$\approx \Phi\left(\frac{B - \delta t \sum_{n=0}^{N-1} \langle \mu(\cdot + z_{t_n})\rangle_Y - \rho_x \sum_{n=0}^{N-1} \langle \sigma^2(\cdot + z_{t_n})\rangle_Y^{\frac{1}{2}} \left(W_{t_{n+1}}^x - W_{t_n}^x\right)}{\sqrt{(1-\rho_x^2)\delta t \sum_{n=0}^{N-1} \langle \sigma^2(\cdot + z_{t_n})\rangle_Y}}\right). \tag{25}$$

Linear Y and Z -average. Third, we approximate the marginal CDF by using an ergodic Y^1 and Z average (abbreviated $\mathrm{erg}_1\mathrm{YZ}$) over their stationary distribution,

$$\bar{f} = \langle\langle f(\cdot + Z)\rangle_Y\rangle_Z = \langle f\rangle_{Y+Z} = \int_{-\infty}^{\infty} f(y)\phi_{Y+Z}(y)\, dy, \tag{26}$$

where ϕ_{Y+Z} is the centered normal density with variance ξ^2/k. Hence, we obtain

$$\mathbb{P}\Big(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y}\Big) \approx \Phi\left(\frac{B - \bar{\mu}T - \rho_x \bar{\sigma} W_T^x}{\sqrt{(1-\rho_x^2)\overline{\sigma^2}T}}\right). \tag{27}$$

Quadratic Y and Z-average. Alternatively, we will use a quadratic ergodic Y^1 and Z average (abbreviated $\mathrm{erg}_2\mathrm{YZ}$) in the stochastic integral to obtain

$$\mathbb{P}\Big(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y}\Big) \approx \Phi\left(\frac{B - \bar{\mu}T - \rho_x \overline{\sigma^2}^{\frac{1}{2}} W_T^x}{\sqrt{(1-\rho_x^2)\overline{\sigma^2}T}}\right). \tag{28}$$

3.3 Approximation of Marginal CDF by a CLT-Type Argument

Here, we introduce an approximation to the marginal CDF in Y^1 (abbreviated *appY*), and hence to the limiting loss function, by adding a correction term from a central limit theorem (CLT). We note that, as $\epsilon \to 0$, the process $(Y_t^1)_{0\le t\le T}$ decorrelates exponentially fast, on the time scale ϵ. Arguing informally with the central limit theorem under strong mixing (see, e.g., [1, Theorem 27.5]), we approximate for small ϵ, conditional on $\mathcal{F}_T^{x,y}$,

$$\frac{\int_0^T \sigma(Y_t^1 + Z_t)\, dW_t^x - \int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y \, dW_t^x}{\sqrt{\int_0^T \Big(\langle \sigma^2(\cdot + Z_t)\rangle_Y - \langle \sigma(\cdot + Z_t)\rangle_Y^2\Big) dt}} \overset{\text{law}}{\approx} W_1, \tag{29}$$

where W_1 is a standard normal random variable. Similarly, for small ϵ and conditional on $\mathcal{F}_T^{x,y}$, we use

$$\int_0^T f(Y_t^1 + Z_t)\, dt \overset{\text{law}}{\approx} \int_0^T \langle f(\cdot + Z_t)\rangle_Y \, dt. \tag{30}$$

Note that, for any $c_0, c_1 \in \mathbb{R}$,

$$\mathbb{E}\Big[\Phi\big(c_0 - c_1 W_1\big)\Big] = \Phi\left(\frac{c_0}{\sqrt{1 + c_1^2}}\right). \tag{31}$$

Combining (16), (18) and (29)–(31) yields

$$\begin{aligned}\mathbb{P}\Big(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y}\Big) &\approx \Phi\left(\frac{B - \int_0^T \langle \mu(\cdot + Z_t)\rangle_Y \, dt - \rho_x \int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y \, dW_t^x}{\sqrt{\int_0^T \langle \sigma^2(\cdot + Z_t)\rangle_Y \, dt - \rho_x^2 \int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y^2 \, dt}}\right) \\ &\approx \Phi\left(\frac{B - \delta t \sum_{n=0}^{N-1} \langle \mu(\cdot + z_{t_n})\rangle_Y - \rho_x \sum_{n=0}^{N-1} \langle \sigma(\cdot + z_{t_n})\rangle_Y \Big(W_{t_{n+1}}^x - W_{t_n}^x\Big)}{\sqrt{\delta t \sum_{n=0}^{N-1} \langle \sigma^2(\cdot + z_{t_n})\rangle_Y - \rho_x^2 \delta t \sum_{n=0}^{N-1} \langle \sigma(\cdot + z_{t_n})\rangle_Y^2}}\right).\end{aligned} \tag{32}$$

3.4 Exponential Ornstein–Uhlenbeck Model

Henceforth, we consider an exponential Ornstein–Uhlenbeck stochastic volatility model for the dynamics of the asset price processes. The drift coefficient is $\mu(y) = -\frac{1}{2}\sigma^2(y)$, whereas the diffusion coefficient is $\sigma(y) = me^y$, see [14]. We do not have a closed-form formula for the conditional CDF is not available under this model.

We substitute the specific coefficients into the above formulae for the conditional CDF and use moment generating functions. From (19), we find an estimate for the conditional (on $\mathcal{F}_T^{y_1}$) marginal (in x) CDF,

$$\mathbb{P}\left(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y,y_1}\right) \approx \Phi\left(\frac{Bm^{-1} + \frac{1}{2}m\delta t \sum_{n=0}^{N-1} e^{2y_{t_n}^1 + 2z_{t_n}}}{\sqrt{(1-\rho_x^2)\delta t \sum_{n=0}^{N-1} e^{2y_{t_n}^1 + 2z_{t_n}}}} - \frac{\rho_x}{\sqrt{1-\rho_x^2}} \frac{\sum_{n=0}^{N-1} e^{y_{t_n}^1 + z_{t_n}} \left(W_{t_{n+1}}^x - W_{t_n}^x\right)}{\sqrt{\delta t \sum_{n=0}^{N-1} e^{2y_{t_n}^1 + 2z_{t_n}}}}\right). \tag{33}$$

From (32), we find an estimate for the approximate conditional CDF,

$$\mathbb{P}\left(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y}\right) \approx \Phi\left(\frac{Bm^{-1}e^{-\frac{\xi^2}{k}(1-\rho_y^2)} + \frac{1}{2}me^{\frac{\xi^2}{k}(1-\rho_y^2)}\delta t \sum_{n=0}^{N-1} e^{2z_{t_n}}}{\sqrt{\left(1-\rho_x^2 e^{-\frac{\xi^2}{k}(1-\rho_y^2)}\right)\delta t \sum_{n=0}^{N-1} e^{2z_{t_n}}}} - \frac{\rho_x}{\sqrt{e^{\frac{\xi^2}{k}(1-\rho_y^2)} - \rho_x^2}} \frac{\sum_{n=0}^{N-1} e^{z_{t_n}} \left(W_{t_{n+1}}^x - W_{t_n}^x\right)}{\sqrt{\delta t \sum_{n=0}^{N-1} e^{2z_{t_n}}}}\right). \tag{34}$$

Y-averages. From (23) and (25), we find an estimate for the conditional CDF with the (linear and quadratic) ergodic Y^1 average,

$$\mathbb{P}\left(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y}\right) \approx \Phi\left(\frac{Bm^{-1}e^{-\frac{\xi^2}{k}(1-\rho_y^2)} + \frac{1}{2}me^{\frac{\xi^2}{k}(1-\rho_y^2)}\delta t \sum_{n=0}^{N-1} e^{2z_{t_n}}}{\sqrt{(1-\rho_x^2)\delta t \sum_{n=0}^{N-1} e^{2z_{t_n}}}} - \frac{\rho_x}{\sqrt{1-\rho_x^2}} \frac{e^{-\lambda\frac{\xi^2}{2k}(1-\rho_y^2)} \sum_{n=0}^{N-1} e^{z_{t_n}} \left(W_{t_{n+1}}^x - W_{t_n}^x\right)}{\sqrt{\delta t \sum_{n=0}^{N-1} e^{2z_{t_n}}}}\right), \tag{35}$$

where $\lambda = 0$ for the quadratic average and $\lambda = 1$ for the linear average.

Y and Z-averages. Finally, from (27) and (28), we find an estimate for the marginal CDF with the (linear or quadratic) ergodic Y^1 and Z average,

$$\mathbb{P}\left(X_T^1 \le B \,|\, \mathcal{F}_T^{x,y}\right) \approx \Phi\left(\frac{Bm^{-1}e^{-\frac{\xi^2}{k}} + \frac{1}{2}me^{\frac{\xi^2}{k}}T}{\sqrt{(1-\rho_x^2)T}} - \frac{\rho_x}{\sqrt{1-\rho_x^2}}\frac{e^{-\lambda\frac{\xi^2}{2k}}W_T^x}{\sqrt{T}}\right). \quad (36)$$

The formulae (33)–(36) indicate that the approximation errors as well as the difference between the approximations will increase with $|\rho_x|$.

3.5 *Pathwise Numerical Tests*

A motivation for considering pathwise tests of the different approximations is the filtering interpretation of the equations.

We fix the time horizon $T = 1$ and the default level $B = -0.1$, and assign the following values to the underlying model parameters:

$$y_0 = 0.2,\ m = 0.1,\ k = 1.0,\ \xi = 0.26,\ \rho_x = 0.9,\ \rho_y = 0.5,\ \rho_{xy} = -0.6; \quad (37)$$

we vary ϵ. We refer to [6, 8] for data that suggest a mean-reversion time of a few days for the S&P500.

To produce the results in Table 1, we fixed the paths for (W^x, W^y), generated by standard sampling of i.i.d. normal increments, and then produced $4 \cdot 10^5$ samples of $W^{y,1}$ to estimate the outer expectation in (16), using the time stepping approximation (19). The number of samples was chosen such that the relative statistical error, estimated as the corrected sample standard deviation of the estimator divided by the value itself, was below 0.15%.

Our tests with different ϵ suggest that the number of time steps should scale with ϵ^{-1} for uniform accuracy. More specifically, for a fraction ϵ of a year, 40 time steps were required for a sufficiently small time-discretization error that matches the statistical error.

The computations were carried out in MATLAB R2016b on a laptop with the following specifications: Intel(R) Core(TM) i7-6700HQ CPU 2.60GHz, 8GB RAM, running Windows 10 (64 bit). The computations below took several hours to compute the 'true' loss, which is why we considered only three 'outer' sample paths. The computation time for the various approximations was negligible as no inner sampling was required. This gain in efficiency is a major motivation for the approximations studied in this paper.

The results are presented in Table 1 and Figure 1.

The approximate conditional CDF (appY), derived from Lyapunov's central limit theorem, provides a good fit to the true conditional CDF for small ϵ across all samples. Figure 1 allows a comparison of this error to an asymptotic behaviour of order $\epsilon^{1/2}$, however, due to the irregular behaviour of the individual path realisations, no definitive conclusions are possible.

For linear averaging of the volatility function in Y^1, $\mathrm{erg}_1\mathrm{Y}$, which gives the correct conditional (on W^x, W^y) expectation of X^1 for $\epsilon \to 0$, the error broadly decreases

Table 1 The marginal CDF (limiting loss function), its approximations and the corresponding relative errors (RE) for different values of ϵ and for three realisations. The number of time steps is $N = 4 \cdot 10^5$ and the relative statistical error is 0.15%

ϵ	Loss	appY	RE(%)	erg_1 Y	RE(%)	erg_2Y	RE(%)	erg_1YZ	RE(%)	erg_2 YZ	RE(%)
10^0	0.49715	0.42303	14.91	0.41542	16.44	0.43350	12.80	0.51886	4.37	0.54713	10.05
10^{-1}	0.53123	0.54974	3.48	0.55471	4.42	0.57538	8.31	0.51886	2.33	0.54713	2.99
10^{-2}	0.67626	0.67967	0.50	0.69624	2.95	0.71826	6.21	0.51886	23.27	0.54713	19.09
10^{-3}	0.58330	0.58041	0.50	0.58835	0.87	0.61049	4.66	0.51886	11.05	0.54713	6.20
10^{-4}	0.47947	0.48005	0.12	0.47805	0.30	0.49815	3.90	0.51886	8.22	0.54713	14.11
10^0	0.45688	0.39515	13.51	0.38492	15.75	0.40177	12.06	0.41543	9.07	0.43981	3.74
10^{-1}	0.46628	0.47136	1.09	0.46849	0.47	0.48729	4.51	0.41543	10.91	0.43981	5.68
10^{-2}	0.48804	0.48556	0.51	0.48412	0.80	0.50405	3.28	0.41543	14.88	0.43981	9.88
10^{-3}	0.32785	0.32987	0.62	0.31403	4.22	0.32807	0.07	0.41543	26.71	0.43981	34.15
10^{-4}	0.41799	0.42107	0.74	0.41327	1.13	0.43136	3.20	0.41543	0.61	0.43981	5.22
10^0	0.17821	0.15721	11.78	0.13417	24.71	0.13932	21.82	0.15003	15.81	0.15806	11.30
10^{-1}	0.13892	0.15375	10.68	0.13075	5.88	0.13572	2.30	0.15003	8.00	0.15806	13.78
10^{-2}	0.17593	0.18456	4.90	0.16152	8.19	0.16803	4.49	0.15003	14.72	0.15806	10.16
10^{-3}	0.25130	0.25282	0.61	0.23195	7.70	0.24213	3.65	0.15003	40.30	0.15806	37.10
10^{-4}	0.31246	0.31196	0.16	0.29477	5.66	0.30801	1.42	0.15003	51.99	0.15806	49.41

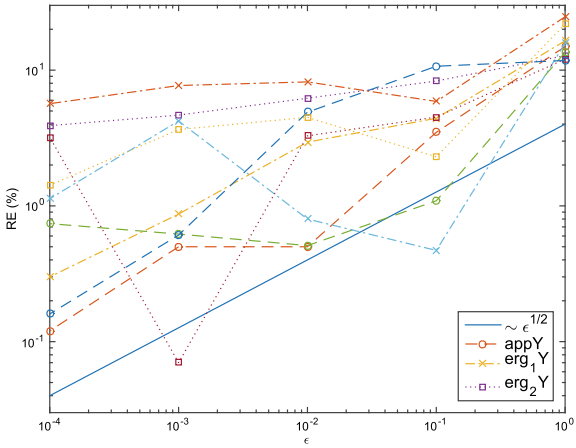


Fig. 1 Double logarithmic plot of the relative errors (RE) of three approximations to the marginal CDF, for three realizations of (W^x, W^y). Shown are the errors of the CLT approximation, appY (circles), and the linear and quadratic Y-averages, erg$_1$Y (crosses) and erg$_2$Y (squares), respectively, for three sample paths

for decreasing ϵ but is significantly larger than in the CLT-based approximation. A similar behaviour is observed for quadratic averaging erg$_2$Y, which gives the correct asymptotic second unconditional moment of X^1.

The approximations based on full (Y and Z) linear and quadratic averages, erg$_1$YZ and erg$_2$YZ, respectively, are independent of ϵ. It is seen from the last columns of Table 1 that they give a poor approximation to the true loss and are therefore not included in Fig. 1 for clarity.

The latter observations are in line with [12, Theorem 2.4] who derive a limiting particle system for $\epsilon \to 0$ in which the averaged squared volatility and a modified correlation coefficient appear. In the SPDE for the limit empirical measure, this is replaced by a linear average and a yet different correlation coefficient. As per [12, Corollary 2.8], this indicates that except for $\rho_y = 0$, convergence is generally only observed in a distributional sense and not strongly.

4 Weak Approximation of Loss Function

In this section, we give an application to basket credit derivatives and analyse numerically the accuracy of the approximations.

Let $a \in [0, 1]$ and consider a call option on the limiting loss function at time T,

$$C_a = \mathbb{E}\left[(L_T - a)^+\right], \tag{38}$$

with fixed default level B. For convenience, we assume that $\rho_x > 0$. This type of payoffs is common in credit derivatives, e.g., in single tranche CDOs [4, 9].

We can compute the call price via (33) by estimating the limiting loss function at time T by Monte Carlo sampling of $W^{y,1}$, and then the outer expectation in (38) by

sampling of W^x and W^y (abbreviated *limCall*). Alternatively, for a large number of firms, we can approximate the call price by

$$C_{N_f,a} = \mathbb{E}\left[(L_{N_f,T} - a)^+\right]$$

and then estimate the expectation by a Monte Carlo average over discrete trajectories of W^x, W^y, $W^{x,1}, \ldots, W^{x,N_f}$, $W^{y,1}, \ldots, W^{y,N_f}$ (abbreviated *firmsCall*). The latter method does not require that we simulate an inner expectation, which can be very expensive, but we lose the smoothness in the loss function.

4.1 Call Price Approximation by Conditional CLT Argument

Here, we approximate the call price using the approximate marginal CDF from (32). We can decompose the Brownian motion W^x as $W^x = \rho_{xy} W^y + \sqrt{1-\rho_{xy}^2}\,\tilde{W}^x$, where W^y and $\tilde{W}^x$ are independent Brownian motions. Let $\mathcal{F}^y$ be the filtration generated by the market Brownian driver W^y. Conditional on the σ-algebra $\mathcal{F}_T^y$,

$$\int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y \, d\tilde{W}_t^x \overset{\text{law}}{=} \sqrt{\int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y^2 \, dt}\; W_1,$$

where W_1 is a standard normal random variable. Hence, the approximate conditional (on $\mathcal{F}_T^y$) law of the limiting loss function at time T is that of $\Phi\big(c_{Y,0} - c_{Y,1} W_1\big)$, where

$$c_{Y,0} = \frac{B - \int_0^T \langle \mu(\cdot + Z_t)\rangle_Y \, dt - \rho_x \rho_{xy} \int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y \, dW_t^y}{\sqrt{\int_0^T \langle \sigma^2(\cdot + Z_t)\rangle_Y \, dt - \rho_x^2 \int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y^2 \, dt}}, \tag{39}$$

$$c_{Y,1} = \rho_x \sqrt{1-\rho_{xy}^2} \sqrt{\frac{\int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y^2 \, dt}{\int_0^T \langle \sigma^2(\cdot + Z_t)\rangle_Y \, dt - \rho_x^2 \int_0^T \langle \sigma(\cdot + Z_t)\rangle_Y^2 \, dt}}. \tag{40}$$

Using a conditioning technique, we can express the call price as

$$C_a = \mathbb{E}\left[\mathbb{E}\left[(L_T - a)^+ \,|\, \mathcal{F}_T^y\right]\right]. \tag{41}$$

Upon noticing that $c_{Y,1} > 0$, we can compute the inner expectation

$$\begin{aligned}\mathbb{E}\Big[(L_T-a)^+\,|\,\mathcal{F}_T^y\Big] &\approx \mathbb{E}\left[\Big(\Phi\big(c_{Y,0}-c_{Y,1}W_1\big)-a\Big)^+\,|\,\mathcal{F}_T^y\right]\\ &= \int_{-\infty}^{w_0}\Big(\Phi\big(c_{Y,0}-c_{Y,1}w\big)-\Phi\big(\Phi^{-1}(a)\big)\Big)\phi(w)\,dw\\ &= \int_{-\infty}^{w_0}\Phi\big(c_{Y,0}-c_{Y,1}w\big)\phi(w)\,dw - a\Phi(w_0),\end{aligned}\tag{42}$$

with ϕ the standard normal PDF and $w_0=\frac{c_{Y,0}-\Phi^{-1}(a)}{c_{Y,1}}$. By [15, formula 10,010.1],

$$\int_{-\infty}^{w_0}\Phi\big(c_{Y,0}-c_{Y,1}w\big)\phi(w)\,dw = \mathrm{BvN}\left(\frac{c_{Y,0}}{\sqrt{1+c_{Y,1}^2}},\,w_0;\,\frac{c_{Y,1}}{\sqrt{1+c_{Y,1}^2}}\right),\tag{43}$$

where the bivariate normal CDF is

$$\mathrm{BvN}\,(h,k;\rho)=\frac{1}{2\pi\sqrt{1-\rho^2}}\int_{-\infty}^{k}\int_{-\infty}^{h}\exp\left(-\frac{x^2-2\rho xy+y^2}{2(1-\rho^2)}\right)dx\,dy.\tag{44}$$

Combining (42)–(43) yields

$$\mathbb{E}\Big[(L_T-a)^+\,|\,\mathcal{F}_T^y\Big]\approx\tag{45}$$

$$\mathrm{BvN}\left(\frac{c_{Y,0}}{\sqrt{1+c_{Y,1}^2}},\,\frac{c_{Y,0}-\Phi^{-1}(a)}{c_{Y,1}};\,\frac{c_{Y,1}}{\sqrt{1+c_{Y,1}^2}}\right)-a\Phi\left(\frac{c_{Y,0}-\Phi^{-1}(a)}{c_{Y,1}}\right)\tag{46}$$

Finally, we discretize the two coefficients, i.e., $c_{Y,0}\approx\bar{c}_{Y,0}$ and $c_{Y,1}\approx\bar{c}_{Y,1}$, where

$$\bar{c}_{Y,0}=\frac{Bm^{-1}e^{-\frac{\xi^2}{k}(1-\rho_y^2)}+\frac{1}{2}me^{\frac{\xi^2}{k}(1-\rho_y^2)}\mathcal{I}-\rho_x\rho_{xy}e^{-\frac{\xi^2}{2k}(1-\rho_y^2)}\mathcal{M}}{\sqrt{\left(1-\rho_x^2e^{-\frac{\xi^2}{k}(1-\rho_y^2)}\right)\delta t\sum_{n=0}^{N-1}e^{2z_{t_n}}}},\ \text{with}\tag{47}$$

$$\mathcal{I}=\sum_{n=0}^{N-1}e^{2z_{t_n}}\delta t,\quad \mathcal{M}=\sum_{n=0}^{N-1}e^{z_{t_n}}\left(W^y_{t_{n+1}}-W^y_{t_n}\right),\quad \textit{and}\quad \bar{c}_{Y,1}=\frac{\rho_x\sqrt{1-\rho_{xy}^2}}{\sqrt{e^{\frac{\xi^2}{k}(1-\rho_y^2)}-\rho_x^2}},$$

and estimate the outer expectation in (41) by Monte Carlo sampling of W^y.

4.2 Call Price Approximation by Ergodic Averages

Y-averages. First, we approximate the call price by employing a linear or quadratic ergodic Y^1 average. Recall from (23) and (25) that

$$L_T \approx \Phi\left(\frac{B - \int_0^T \langle \mu(\cdot + Z_t)\rangle_Y \, dt - \rho_x \int_0^T \langle \sigma^{2-\lambda}(\cdot + Z_t)\rangle_Y^{\frac{1}{2-\lambda}} \, dW_t^x}{\sqrt{(1-\rho_x^2)\int_0^T \langle \sigma^2(\cdot + Z_t)\rangle_Y \, dt}}\right). \tag{48}$$

Proceeding as before, we deduce that

$$\mathbb{E}\Big[(L_T - a)^+ \,|\, \mathcal{F}_T^y\Big] \approx \mathrm{BvN}\left(\frac{c_{Y,2}}{\sqrt{1+c_{Y,3}^2}}, \frac{c_{Y,2} - \Phi^{-1}(a)}{c_{Y,3}}; \frac{c_{Y,3}}{\sqrt{1+c_{Y,3}^2}}\right) - a\Phi\left(\frac{c_{Y,2} - \Phi^{-1}(a)}{c_{Y,3}}\right), \tag{49}$$

where

$$c_{Y,2} = \frac{B - \int_0^T \langle \mu(\cdot + Z_t)\rangle_Y \, dt - \rho_x \rho_{xy} \int_0^T \langle \sigma^{2-\lambda}(\cdot + Z_t)\rangle_Y^{\frac{1}{2-\lambda}} \, dW_t^y}{\sqrt{(1-\rho_x^2)\int_0^T \langle \sigma^2(\cdot + Z_t)\rangle_Y \, dt}}, \tag{50}$$

$$c_{Y,3} = \frac{\rho_x\sqrt{1-\rho_{xy}^2}}{\sqrt{1-\rho_x^2}} \sqrt{\frac{\int_0^T \langle \sigma^{2-\lambda}(\cdot + Z_t)\rangle_Y^{\frac{2}{2-\lambda}} \, dt}{\int_0^T \langle \sigma^2(\cdot + Z_t)\rangle_Y \, dt}}. \tag{51}$$

As before, we discretize the two coefficients, i.e., $c_{Y,2} \approx \bar{c}_{Y,2}$ and $c_{Y,3} \approx \bar{c}_{Y,3}$, where

$$\bar{c}_{Y,2} = \frac{Bm^{-1}e^{-\frac{\xi^2}{k}(1-\rho_y^2)} + \frac{1}{2}me^{\frac{\xi^2}{k}(1-\rho_y^2)}\mathcal{I} - \rho_x\rho_{xy}e^{-\lambda\frac{\xi^2}{2k}(1-\rho_y^2)}\mathcal{M}}{\sqrt{(1-\rho_x^2)\delta t \sum_{n=0}^{N-1} e^{2z_{t_n}}}}, \tag{52}$$

$$\bar{c}_{Y,3} = \frac{\rho_x\sqrt{1-\rho_{xy}^2}}{\sqrt{1-\rho_x^2}} e^{-\lambda\frac{\xi^2}{2k}(1-\rho_y^2)}, \tag{53}$$

and estimate the outer expectation in (41) by a sample average over W^y.

Y and Z-averages. Last, we approximate the call price by linear and quadratic ergodic Y^1 and Z average. Using (27) and (28), we can deduce in a similar fashion

$$C_a \approx \mathrm{BvN}\left(\frac{c_0}{\sqrt{1+c_1^2}}, \frac{c_0-\Phi^{-1}(a)}{c_1}; \frac{c_1}{\sqrt{1+c_1^2}}\right) - a\Phi\left(\frac{c_0-\Phi^{-1}(a)}{c_1}\right), \tag{54}$$

$$\text{where} \quad c_0 = \frac{Bm^{-1}e^{-\frac{\xi^2}{k}} + \frac{1}{2}me^{\frac{\xi^2}{k}}T}{\sqrt{(1-\rho_x^2)T}}, \qquad c_1 = \frac{\rho_x}{\sqrt{1-\rho_x^2}}e^{-\lambda\frac{\xi^2}{2k}}. \tag{55}$$

4.3 Expected Loss

In the special case of a linear payoff ($a = 0$), the call price is simply the expected limiting loss function at time T (abbreviated *expLoss*). Using a conditioning technique and (15), we can write the expected loss as

$$\mathbb{E}[L_T] = \mathbb{E}\left[\mathbb{E}\left[\mathbb{1}_{X_T^1 \le B} \,|\, \mathcal{F}_T^{x,y}\right]\right] = \mathbb{P}\left(X_T^1 \le B\right) = \mathbb{E}\left[\mathbb{P}\left(X_T^1 \le B \,|\, \mathcal{F}_T^{y,y_1}\right)\right]. \tag{56}$$

From (3) we deduce that

$$\mathbb{E}[L_T] = \mathbb{E}\left[\Phi\left(\frac{B - \int_0^T \mu(Y_t^1 + Z_t)\,dt - \rho_x\rho_{xy}\int_0^T \sigma(Y_t^1 + Z_t)\,dW_t^y}{\sqrt{(1-\rho_x^2\rho_{xy}^2)\int_0^T \sigma^2(Y_t^1 + Z_t)\,dt}}\right)\right], \tag{57}$$

which can be estimated by a Monte Carlo average over samples of W^y and $W^{y,1}$. Hence, this provides a much faster method in the special case of a linear payoff.

4.4 Numerical Tests

We perform numerical tests for the weak errors with the different approximations. We fix the time horizon $T = 1$ and the default level $B = -0.1$ as in Sect. 3, and assign the same values to the underlying model parameters as in (37). Furthermore, we fix $\epsilon = 4 \cdot 10^{-3}$ as in [5], a choice which corresponds to a mean-reversion time of 1.5 days, as observed from S&P500 data (see [8]).

The number of samples for the outer expectations in, e.g., (41), was $1.2 \cdot 10^6$ and gave a small statistical error, estimated as the corrected sample standard deviation of the estimator divided by the value itself, of 0.15% for the call price and 0.05% for the approximations and the expected loss. We have verified numerically that the errors associated with the number of time steps N and the number of firms N_f from Table 2 match the statistical errors.

Table 2 The call option price, its approximations and the corresponding relative errors (RE) for 3 different strikes $a \in \{0.00, 0.05, 0.10\}$. The number of time steps is $N = 10^4$, the number of firms is $N_f \in \{5, 150, 100\}$—each value corresponds to one of the 3 strikes—and the relative statistical errors are 0.15% for the call price and 0.05% for the approximations and the expected loss

	Strike = 0.00		Strike = 0.05		Strike = 0.10	
Method	Price	RE (%)	Price	RE (%)	Price	RE (%)
expLoss	0.18835	–	–	–	–	–
firmsCall	0.18843	0.05	0.16170	–	0.14132	–
appY	0.18878	0.23	0.16155	0.09	0.14078	0.38
erg_1 Y	0.18390	2.36	0.15860	1.92	0.13941	1.35
erg_2 Y	0.18872	0.20	0.16342	1.06	0.14410	1.97
erg_1 YZ	0.18262	3.04	0.15724	2.76	0.13789	2.42
erg_2 YZ	0.18912	0.41	0.16376	1.27	0.14423	2.06

We infer from the data in Table 2 that $\epsilon = 4 \cdot 10^{-3}$ gives a very small appY-approximation error throughout. Squared averaging (conditional on Z or unconditional), where the first two moments of the X process are matched, results in a very good approximation for a linear payoff, but in a worse approximation than the linear average for a non-linear payoff.

5 Conclusions

It has recently been shown theoretically in [12] that large pool models of processes with fast mean-reverting stochastic volatility may be approximated by one-dimensional models with constant, averaged model parameters. The limit as the mean-reversion speed goes to infinity is generally only attained in a distributional, but not in a strong sense.

We show in this paper how such averaged equations can be implemented numerically, but also observe that the approximation quality is poor in both the strong and the weak sense in cases of interest.

The main finding of the paper is an improved approximation obtained by a central limit theorem argument, which leads to consistently good accuracy both in a pathwise sense conditional on common noise, and in a weak sense when considering expected nonlinear functionals of the solution. A theoretically rigorous analysis of this empirically improved estimator will be the topic of future research.

References

1. Billingsley, P.: Probability and Measure. Wiley, New York (1995)
2. Bujok, K., Hambly, B., Reisinger, C.: Multilevel simulation of functionals of Bernoulli random variables with application to basket credit derivatives. Methodol. Comput. Appl. Probab. **17**, 579–604 (2015)
3. Bujok, K., Reisinger, C.: Numerical valuation of basket credit derivatives in structural jump-diffusion models. J. Comput. Fin. **15**, 115–158 (2012)
4. Bush, N., Hambly, B., Haworth, H., Jin, L., Reisinger, C.: Stochastic evolution equations in portfolio credit modelling. SIAM J. Financ. Math. **2**, 627–664 (2011)
5. Dobson, P.: Using two time scales to accurately approximate the behaviour of a large pool of stochastic volatility models. Master's thesis, University of Oxford (2015)
6. Fouque, J.P., Papanicolaou, G., Sircar, R.: Mean-reverting stochastic volatility. Int. J. Theor. Appl. Finance **3**(1), 101–142 (2000)
7. Fouque, J.P., Papanicolaou, G., Sircar, R., Solna, K.: Multiscale stochastic volatility asymptotics. Multiscale Mod. Sim. **2**(1), 22–42 (2003)
8. Fouque, J.P., Papanicolaou, G., Sircar, R., Solna, K.: Short time-scale in S&P500 volatility. J. Comput. Fin. **6**, 1–24 (2003)
9. Giles, M., Reisinger, C.: Stochastic finite differences and multilevel Monte Carlo for a class of SPDEs in finance. SIAM J. Financ. Math. **3**, 572–592 (2012)
10. Hambly, B., Kolliopoulos, N.: Stochastic evolution equations for large portfolios of stochastic volatility models. SIAM J. Financ. Math. **8**, 962–1014 (2017)
11. Hambly, B., Kolliopoulos, N.: Stochastic PDEs for large portfolios with general mean-reverting volatility processes (2019). arXiv:1906.05898
12. Hambly, B., Kolliopoulos, N.: Fast mean-reversion asymptotics for large portfolios of stochastic volatility models. Fin. Stochast. **24**, 757–794 (2020)
13. Ledger, S.: Sharp regularity near an absorbing boundary for solutions to second order SPDEs in a half-line with constant coefficients. Stoch. Partial Diff. Equ.: Anal. Comput. **2**, 1–26 (2014)
14. Masoliver, J., Perelló, J.: Multiple time scales and the exponential Ornstein-Uhlenbeck stochastic volatility model. Quant. Fin. **6**, 423–433 (2006)
15. Owen, D.: A table of normal integrals. Comm. Stat.: Simul. Comput. B **9**, 389–419 (1980)
16. Shiryaev, A.: Probability. Springer, Berlin (1996)

Generating From the Strauss Process Using Stitching

Mark Huber

Abstract The Strauss process is a point process with unnormalized density with respect to a Poisson point process of rate λ, where each pair of points within a specified distance r of each other contributes a factor $\gamma \in [0, 1]$ to the density. Basic acceptance-rejection works spectacularly poorly for this problem, which is why several other perfect simulation methods have been developed. These methods, however, also work poorly for reasonably large values of λ. *Recursive Acceptance Rejection Stitching* is a new method that works much faster, allowing the simulation of point processes with values of λ much larger than ever before.

Keywords Perfect simulation · Spatial process · Acceptance rejection

1 Introduction

Repulsive point processes arise as models of spatial data where points lie farther apart from one another than would be expected if they were independent. For instance, Glass and Tobler [2] studied the locations of cities on a plain in Spain and Strand [16] looked at the locations of a certain species of trees in a forest. In both instances, the points appear to repulse one another, and appear farther apart than would be seen in a basic Poisson point process.

To fit a statistical model to this type of data, Strauss [17] introduced what today is known as the *Strauss process*. This is a point process that has a density with respect to an underlying Poisson point process that penalizes configurations where pairs of points are too close to one another. Such a model can be used to model data from repulsive point processes [8].

As with most such unnormalized density models, there is no known efficient way to find the normalizing constant of the model for various values of the parameters. Hence to find something like a maximum likelihood estimate or a method of moments estimate, it is necessary to take a Monte Carlo approach. Early methods built con-

M. Huber (✉)
Claremont McKenna College, 850 Columbia AV, Claremont, CA 91711, USA
e-mail: mhuber@cmc.edu

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_12

tinuous time Markov chains [14] which could obtain approximate samples, but the mixing time of such chains was often unknown.

More recently, *perfect simulation* algorithms such as coupling from the past [15] were developed that could draw samples exactly from the stationary distribution of Markov chains. This type of algorithm employed a random number of recursive calls in order to draw the samples. The *dominated coupling from the past* method was created by Kendall and Møller [10] to draw from the stationary distribution of continuous time Markov chains.

However, this algorithm had limits on its effectiveness. One of the parameters of the Strauss process was λ, which controls the number of points. For small values of λ, these types of algorithms ran in polynomial time, but for larger values exponential time was needed. If the data was best fitted by a Strauss process with λ greater than the algorithm could handle, the results would be unreliable.

New types of perfect simulation algorithms not based on Markov chains, such as the partial rejection sampling (PRS) method of Guo and Jerrum [7] appeared, but they did not increase the λ values where the algorithm was effective.

In this work, a completely different approach is used. It is still a perfect simulation algorithm that returns draws exactly from the Strauss process, but by careful management of how draws are taken, can be effective in ranges of λ not possible before.

1.1 The Strauss Process

Given a space $S \subset \mathbb{R}^n$ of nonzero finite Lebesgue measure, say that $X \subset S$ is a *point process* if it contains a finite number of points with probability 1.

A point process X is a *Poisson point process of rate* λ *over* S if the number of points in X has a Poisson distribution with mean equal to λ times the Lebesgue measure of S, and given the number of points in X, each is uniformly distributed over S. The new method presented here works on Poisson point processes with more general rate functions, but for simplicity of presentation, here the rate will be assumed to be a constant λ over S. The set S need not be convex or connected.

For a point process $X \subset S$ and positive constant r, let $c_r(X)$ be the number of pairs of distinct points in X that are at most distance r apart. For $\gamma \in [0, 1]$, let

$$f_{\gamma,r}(x) = \gamma^{c_r(x)}. \tag{1}$$

A point process with (unnormalized) density $f_{\gamma,r}$ with respect to the underlying measure that is a Poisson point process with rate λ over S is a Strauss process [17]. Because $\gamma \leq 1$, this density penalizes point process that have many points within distance r of each other. This density can also be written as a product:

$$f(x) = \prod_{\{x_i,x_j\}\subseteq x} [\gamma \cdot \mathbb{I}(\text{dist}(x_i, x_j) \leq r) + 1 \cdot \mathbb{I}(\text{dist}(x_i, x_j) > r)]. \tag{2}$$

Here $\mathbb{I}$ is the usual indicator function that evaluates to 1 if the argument is true and is 0 otherwise.

Say that a density is a *bounded factor density* if it consists of factors each of which is at most 1. For such a density, the *acceptance-rejection* (AR) method can be used to generate samples exactly from the target distribution. This method was the first method for generating exactly from the Strauss process. The general AR protocol goes back to [13]. More recently, new perfect simulation methods for the Strauss process have been developed. These include:

- Dominated coupling from the past (DCFTP) [9–11].
- Birth-death-swap with bounding chains (BDS) [5].
- Partial rejection sampling (PRS) [7] (when $\gamma = 0$).

See [4, 6] for more detail and the theory underlying these methods. In particular, DCFTP, BDS, and PRS all rely on the process being *locally stable*. A density f is locally stable if for any set of points $x \subset S$ and any point a in S, there is a constant K such that $f(x \cup \{a\}) \leq K f(x)$ (see [10].) Approaches that require local stability will be referred to as *local methods*.

The running time of AR is exponential in λ and the size of the point space S. The running time of local methods tend to be polynomial in the size of S when λ lies below a certain threshold (the critical value) and then exponential above that threshold. This makes generating from the Strauss process difficult for high values of λ.

In this work, a new method for generating from the Strauss process is presented. Like generic AR the new method has an exponential running time in λ, but the rate of exponential growth is much smaller in the size of the point space S. Therefore, the rate of the exponential is much lower than both AR and local methods past their critical point.

The result is an algorithm that allows generation of Strauss processes over (λ, γ, S) triples that were computationally infeasible before. For instance, Fig. 1 illustrates such a process with $\lambda = 200$, $\gamma = 0$, and $r = 0.15$. Using the convention that $0^0 = 1$, note that the Strauss process for $\gamma = 0$ reduces to the uniform density over configurations where the distance between any two points is greater than r.

The rest of the paper is organized as follows. The next section presents the new stitching algorithm, and contains the proof that the algorithm is correct. Section 4 then gives numerical results on the running time. Section 5 then concludes.

2 Acceptance Rejection and Stitching

For a given target density h, let $h_S(x) = h(x \cap S)$ be the density applied to the points in x that fall into S. Given that h is an unnormalized bounded factor density with $h \leq 1$ for all S together with underlying probability measure μ, the general AR algorithm begins by drawing a point process X from the reference measure μ over S. With probability $h_S(X)$, X is accepted as coming from density h_S with respect to μ.

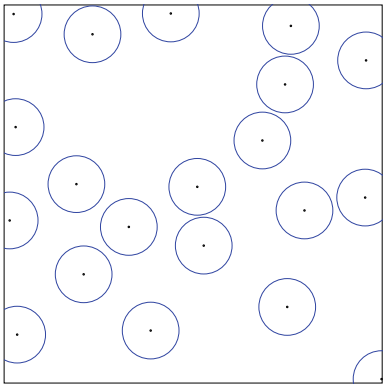


Fig. 1 This represents a Strauss process over $S = [0, 1] \times [0, 1]$ with $\gamma = 0$, $\lambda = 200$, and $r = 0.15$. The black dots represent the actual locations of the points. The fact that $\gamma = 0$ means that no two points are located within distance 0.15 of each other. Around each black dot is a blue circle of radius $0.15/2$ in order to illustrate that in fact all of these points are this far apart. This also shows why this model is also known as a *hard-disks* model

Otherwise, the point process X is rejected as a sample. If rejection occurs, then recursion is used and the acceptance rejection algorithm calls itself to find a new sample Y which is then returned as the draw. For speed purposes a while or repeat loop is typically used instead of recursive calls for basic AR.

The recursive acceptance rejection stitching (RARS) algorithm is different. Instead of generating a sample over the entire space of points S, first S is partitioned into two regions S_1 and S_2. The algorithm then recursively draws X_1 from h_{S_1} and X_2 from h_{S_2}. Next suppose that the target h_S can be factored as follows:

$$h_S(x) = h_{S_1}(x \cap S_1) h_{S_2}(x \cap S_2) h_{S_1,S_2}(x),$$

where each factor is at most 1.

The first two factors have the same form as the original h_S, just over smaller spaces. Therefore, recursive calls to our algorithm can be used to sample draws from these densities. The final factor stitches the two factors together to result in the overall density. In order to make sure that this factor appears in the density of the output of the algorithm, a rejection step will be implemented. Given a proposed configuration found by combining the output of the recursive calls over S_1 and S_2, the result will be accepted with probability proportional to the third h_{S_1,S_2} density evaluated at the proposed configuration.

This idea leads to the following probabilistic recursive algorithm, which has three inputs. First is the target density h, next the underlying probability measure μ which is easy to sample from, and finally the space S where the points are located. The output of the algorithm will be a draw from the density h_S with respect to μ.

RARS(h, μ, S)

1. Draw X from μ with point space S.
2. Draw U_1 uniform over [0, 1].
3. If $U_1 \leq h_S(X)$, then return X and quit.
4. Partition S into (S_1, S_2).
5. Draw X_1 using RARS(h, μ, S_1), draw X_2 using RARS(h, μ, S_2).
6. Draw U_2 uniformly from [0, 1].
7. If $U_2 \leq h_{S_1,S_2}(X_1 \cup X_2)$ then return $X_1 \cup X_2$.
8. Else draw Y from RARS(h, μ, S). Return Y and exit.

The proof uses *The Fundamental Theorem of Perfect Simulation* (FTPS) [6] which gives two conditions under which the output of a probabilistic recursive algorithm $\mathcal{A}$ comes from the target distribution. These conditions are as follows.

1. The algorithm $\mathcal{A}$ must terminate with probability 1 on all inputs.
2. The algorithm must be *locally correct*. Consider an algorithm $\mathcal{A}'$ where the recursive calls in $\mathcal{A}$ are replaced with oracles that generate from the correct distribution. If $\mathcal{A}'$ has output that provably comes from the correct distribution, say that $\mathcal{A}'$ is locally correct.

For the following lemmas, let

$$Z_h = \int h_S(x)\, d\mu(x)$$

be the normalizing constant for the unnormalized density h_S.

Lemma 1 *If $Z_h > 0$, then* RARS(h, μ, S) *terminates in finite time with probability 1 regardless of the choice of partition at line 4.*

Proof Let $r(p)$ be the supremum over the expected number of times X is generated over all choices of S_1, S_2, and h when the probability X is accepted in line 3 is p. Our goal will be to bound $r(p)$ in terms of $p > 0$.

Let p_1 be the probability that X is accepted in the recursive call over S_1, p_2 the acceptance probability over S_2, and p_3 the probability that (X_1, X_2) is accepted in line 7.

Because h_S factors into h_{S_1}, h_{S_2} and h_{S_1,S_2}, $p = p_1 p_2 p_3$. There is always at least one draw of X in any call, followed by a $1 - p$ chance of two recursive calls, followed by a $1 - p_3$ chance of a third recursive call. Hence

$$r(p) \leq 1 + (1 - p)[r(p_1) + r(p_2) + (1 - p_3)r(p)]. \tag{3}$$

This holds for all $p' \geq p$, so letting $w = \sup_{p' \in [p,1]} r(p)$ gives

$$w \leq 1 + (1 - p)[3w]. \tag{4}$$

An easy calculation then gives for $p \geq 3/4$, $r(p) \leq w \leq 4$.

This forms the base case for an induction proof of the following fact: For all $i \in \{0, 1, 2, \ldots\}$, if $p \geq (3/4)(1-p)^i$, then $r(p)$ is finite.

Consider the induction step: suppose for all $p \geq (3/4)(1-p)^i$, there is a finite M such that $r(p) \leq M$. Consider $i+1$, and assume $p \geq (3/4)(1-p)^{i+1}$.

If $p_1 \geq (3/4)(1-p)^i$ and $p_2 \geq (3/4)(1-p)^i$, then

$$r(p) \leq 1 + (1-p)[M + M + (1-p_3)r(p)], \tag{5}$$

and $r(p) \leq (1 + (1-p)2M)/(1-(1-p)(1-p_3))$, completing the induction.

Note that if $p_3 < 1-p$, then $p_1 > p/p_3 = (3/4)(1-p)^i$ and $p_2 > p/p_3 = (3/4)(1-p)^i$ and so the induction step also holds in this case.

It cannot hold that both p_1 and p_2 are less than $(3/4)(1-p)^i$, as that would make $p < (9/16)(1-p)^{2i} < (3/4)(1-p)^{i+1}$. Hence the only case that remains to consider is if $p_3 > 1-p$ and exactly one of p_1 or p_2 (say p_1 without loss of generality) is less than $(3/4)(1-p)^i$.

If $p_3 > 1-p$, then $(1-p_3) < p$, and by the induction hypothesis

$$r(p) \leq 1 + (1-p)[r(p) + M + p \cdot r(p)], \tag{6}$$

which gives $r(p) \leq (1 + (1-p)M)/p^2$, completing the induction.

Since $p > 0$, there is some i such that $p \geq (3/4)(1-p)^i$, and so $r(p)$ is finite for all $p > 0$. □

Lemma 2 *If $Z_h > 0$, then* `RARS`(h, μ, S) *terminates in finite time with probability 1 with output distributed as unnormalized density h_S with respect to μ over S.*

Proof The algorithm terminates with probability 1 by the previous lemma. Hence by the FTPS, it is only necessary to show that the algorithm is locally correct.

Let $\mathcal{A}'$ be the algorithm where lines 5 and 8 are replaced with oracles drawing from the correct distributions. In particular, Y is a draw from μ restricted to point space S. For any measurable B, note

$$\int_B h_S(x)\, d\mu(x) = \frac{Z_h}{Z_h} \int_B h_S(x)\, d\mu(x) = Z_h \mathbb{P}(Y \in B).$$

Fix a measurable set A, and let W be the output of $\mathcal{A}'$. Then the chance the output is in A can be broken down into the probability of three disjoint events e_1, e_2, and e_3, representing acceptance at line 3, rejection at line 3 and acceptance at line 7, or rejection at lines 3 and 7 and $Y \in A$ respectively. That is, $\mathbb{P}(W \in A) = \mathbb{P}(e_1) + \mathbb{P}(e_2) + \mathbb{P}(e_3)$ where

$$\begin{aligned}
e_1 &= (X \in A, U_1 \leq h_S(X)) \\
e_2 &= \left(U_1 > h_S(X), X_1 \cup X_2 \in A, U_2 \leq h_{S_1,S_2}(X_1 \cup X_2)\right) \\
e_3 &= \left(U_1 > h_S(X), U_2 > h_{S_1,S_2}(X_1 \cup X_2), Y \in A\right).
\end{aligned}$$

From basic AR theory (see for instance [1, 3, 12]),

$$\mathbb{P}(e_1) = \mathbb{P}(Y \in A) Z_h. \tag{7}$$

The chance that X is not accepted at line 3 is

$$\mathbb{P}(U > h_S(X)) = 1 - \mathbb{P}(U \leq h_S(X)) = 1 - \int h_S(x)\, d\mu = 1 - Z_h. \tag{8}$$

Since (X, U_1) and (X_1, X_2, U_2) are independent:

$$\begin{aligned}
\mathbb{P}(e_2) &= \mathbb{P}(U_1 > h_S(X))\mathbb{P}(X_1 \cup X_2 \in A, U_2 > h_{S_1,S_2}(X_1 \cup X_2)) \\
&= (1 - Z_h) \int_{x_1 \cup x_2 \in A} h_{S_1}(x_1) h_{S_2}(x_2) h_{S_1,S_2}(x_1, x_2)\, d\mu \\
&= (1 - Z_h) \int_{x_1 \cup x_2 \in A} h_S(x_1 \cup x_2)\, d\mu \\
&= (1 - Z_h) Z_h \mathbb{P}(Y \in A).
\end{aligned}$$

Similarly, using independence of the pieces of the last term, the chance that we reject twice and then the recursive call lands in A is

$$\mathbb{P}(e_3) = (1 - Z_h)(1 - Z_h)\mathbb{P}(Y \in A). \tag{9}$$

Putting these terms together gives

$$\begin{aligned}
\mathbb{P}(W \in A) &= \mathbb{P}(Y \in A)[Z_h + (1 - Z_h)Z_h + (1 - Z_h)^2] \\
&= \mathbb{P}(Y \in A)
\end{aligned}$$

which completes the proof of correctness. □

In some cases, it is possible to know when h is easy to sample from using AR, at which point, one can substitute basic AR in for line 3. In AR for the Strauss process, acceptance occurs with probability 1 when there are no points in the draw. Hence for S small enough that $\lambda S < 1$, there is at least an $\exp(-1)$ chance of accepting. The criterion for what is easy will vary from problem to problem.

RAR-stitch-base (h, μ, S)

1. For (h, S) easy, draw Z using AR. Return Z.
2. Partition S into S_1 and S_2.
3a. Draw X_1 using `RAR-stitch-base` (h, μ, S_1).
3b. Draw X_2 using `RAR-stitch-base` (h, μ, S_2).
4. Draw U_2 uniformly from [0, 1].

5. If $U_2 \leq h_{S_1,S_2}(X_1 \cup X_2)$ then return $X_1 \cup X_2$.
6. Else draw Y from `RAR-stitch-base` (h, μ, S).

Lemma 3 *If $Z_h > 0$, then* `RAR-stitch-base`(h, μ, S) *terminates in finite time with probability 1 with output distributed as unnormalized density h with respect to μ over S.*

Proof The proof follows the same outline as for `RARS`(h, μ, S). □

3 Stitching in Practice

To illustrate stitching in practice, consider the Strauss process and the Ising model.

3.1 The Strauss Process

The Strauss density is determined by the parameters S, λ, r, and γ.

Given a process X_1 over S_1 and X_2 over S_2, $h_{S_1,S_2}(X_1 \cup X_2)$ is γ raised to the number of pairs of points (x_1, x_2) where $x_1 \in S_1$ and $x_2 \in S_2$ that are within distance r of each other. This gives the following algorithm.

`Strauss-AR-stitch-base` (λ, r, γ, S)

1. If λ times the Lebesgue measure of S is at most 5, draw X using AR, and return X.
2. Partition S into S_1 and S_2.

3a. Draw X_1 using `Strauss-AR-stitch-base` $(\lambda, r, \gamma, S_1)$.

3b. Draw X_2 using `Strauss-AR-stitch-base` $(\lambda, r, \gamma, S_2)$.

4. Draw U_2 uniformly from [0, 1]. Let c be the number of $a_i \in X_1$ and $b_j \in X_2$ such that $\text{dist}(a_i, b_j) \leq r$.
5. If $U_2 \leq \gamma^c$ then return $X_1 \cup X_2$.
6. Else return a draw from `Strauss-AR-stitch-base` (λ, r, γ, S).

3.2 The Ising Model

In the Ising model (and its extension, the Potts model), each vertex of a graph is given a label from a color set. In the ferromagnetic model, edges of the graph are penalized by $\exp(-2\beta)$ (for $\beta > 0$ a constant) when the two edges of the set are colored differently. The reference measure is uniform over all colorings of the vertices.

In other words, the density for a graph with edge set E is

$$f(x) = \prod_{\{i,j\} \in E} [\exp(-2\beta)\mathbb{I}(x(i) \neq x(j)) + \mathbb{I}(x(i) = x(j))] \tag{10}$$

with respect to counting measure over all colorings of the vertices of the graph.

A partition of a vertex set of a graph is called a *cut*. The stitching needs only check edges which *cross the cut*, meaning that the endpoints of the edges lie in different halves of the cut. The density (and reference measure) are determined by β, the edge set E, and the vertex set V.

`Ising-AR-stitch-base` (β, E, V)

1. If $V = \{v\}$, then choose $X(v)$ uniformly from the set of colors, return X.
2. Partition V into V_1 and V_2.

3a. Draw X_1 using `Ising-AR-stitch-base` (β, E, V_1).

3b. Draw X_2 using `Ising-AR-stitch-base` (β, E, V_2).

4. Draw U_2 uniformly from $[0, 1]$. Let c be the number of $i \in V_1$ and $j \in V_2$ such that $x(i) \neq x(j)$.
5. If $U_2 \leq \exp(-2\beta c)$ then return (X_1, X_2).
6. Else return a draw from `Ising-AR-stitch-base` (β, E, V).

4 Numerical Results

For the Ising model, there are effective methods to perfectly sample for β above and below the critical temperature [15], so only the Strauss process is considered here.

In order to evaluate the running time behavior of various algorithms for generating from the Strauss process, timings were run on $S = [0, 1] \times [0, 1]$ for basic AR, the PRS method of [7], and stitching. AR is always exponential in λ, while PRS stays polynomial in λ before moving to exponential in λ past a certain threshold. Recursive Acceptance Rejection Stitching is also exponential in λ, but at a much smaller rate (Fig. 2).

Whenever one is constructing an RARS algorithm, a choice must be made how to partition the state space. Here the choice made was as follows. Given a rectangular space S, partition into two rectangles by dividing the space in half using a line parallel to the shorter sides.

A plot of the log of the timings shows the exponential nature of the growth. The original `AR` algorithm quickly becomes exponential in λ, while PRS stays polynomial until the critical point where it switches over to exponential behavior. RARS also appears to be polynomial before turning exponential, but the slope of the log line is much lower than that of AR and PRS, allowing for sampling from much higher values of λ (Fig. 3).

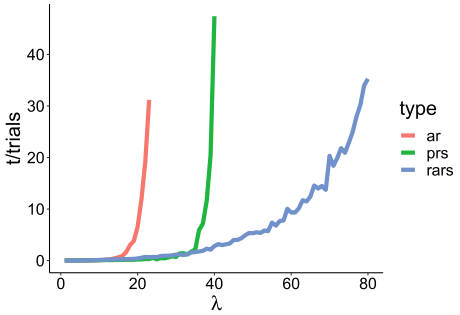


Fig. 2 Timings of Acceptance Rejection, Partial Rejection Sampling, and Recursive Acceptance Rejection with Stitching for varying λ over $S = [0, 1] \times [0, 1]$

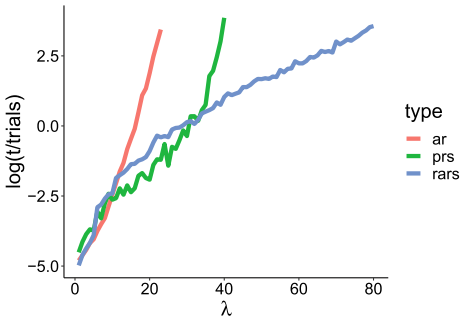


Fig. 3 Log timings of Acceptance Rejection, Partial Rejection Sampling, and Recursive Acceptance Rejection with Stitching for varying λ over $S = [0, 1] \times [0, 1]$.

4.1 Code

The code for the numerical examples above can be found in Sect. 4 of https://arxiv.org/pdf/2012.08665.pdf.

5 Conclusion

Stitching is a simple to implement algorithm that has an exponential running time with a rate far lower than either acceptance rejection or various local methods. It applies to any density that can be factored into three pieces where two of the pieces can be sampled from recursively, and the third piece is at most 1. In particular, this applies to point processes such as the Strauss process. Experimentally, it has been found that this algorithm can generate exact instances of the Strauss process over values of the parameter space that were previously unobtainable in a reasonable amount of time.

References

1. Devroye, L.: Non-uniform Random Variate Generation. Springer, Berlin (1986)
2. Glass, L., Tobler, W.: Uniform distribution of objects in a homogeneous field: Cities on a plain. Nature **233**, 67–68 (1971)
3. Hörmann, W., Leydold, J., Derflinger, G.: Automatic Nonuniform Random Variate Generation. Springer, Berlin (2004)
4. Huber, M.: Spatial point processes. In: Brooks, S., Gelman, A., Jones, G., Meng, X. (eds.) Handbook of MCMC, pp. 227–252. Chapman & Hall/CRC Press (2011)
5. Huber, M.: Spatial birth-death swap chains. Bernoulli **18**(3), 1031–1041 (2012). ArXiv:1006.5934
6. Huber, M.L.: Perfect Simulation. No. 148 in Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press (2015)
7. Jerrum, M., Guo, H.: Perfect simulation of the hard disks model by partial rejection sampling. Annales de l'Institut Henri Poincaré D (AIHPD) (2019)
8. Kelly, F., Ripley, B.D.: A note on Srauss's model for clustering. Biometrika **63**(2) (1976)
9. Kendall, W.S.: Perfect simulation for the area-interaction point process. In: Proceedings of the Symposium on Probability Towards the Year 2000 (1995)
10. Kendall, W.S., Møller, J.: Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. Adv. Appl. Prob. **32**, 844–865 (2000)
11. Kendall, W.S., Thönnes, E.: Perfect simulation in stochastic geometry. Pattern Recognit. **32**, 1569–1586 (1999)
12. P. L'Ecuyer. Random number generation. In: Gentle, J.E., Haerdle, W., Mori, Y. (eds.) Handbook of Computational Statistics, 2nd edn. pp. 35–71. Springer, Berlin (2012)
13. von Neumann, J.: Various techniques used in connection with random digits. In: Monte Carlo Method, Applied Mathematics Series 12. National Bureau of Standards, Washington, D.C. (1951)
14. Preston, C.: Spatial birth-and-death processes. Bull. Inst. Int. Stat. **46**(2), 371–391 (1977)
15. Propp, J.G., Wilson, D.B.: Exact sampling with coupled Markov chains and applications to statistical mechanics. Random Struct. Algorithms **9**(1–2), 223–252 (1996)
16. Strand, L.: A model for stand growth. In: IUFRO Third Conference Advisory Group of Forest Statisticians, vol. 72, p. 3. INRA, Institut National de la Recherche Agronomique Paris (1972)
17. Strauss, D.J.: A model for clustering. Biometrika **63**, 467–475 (1975)

A Note on Transformed Fourier Systems for the Approximation of Non-periodic Signals

Robert Nasdala and Daniel Potts

Abstract A variety of techniques have been developed for the approximation of non-periodic functions. In particular, there are approximation techniques based on rank-1 lattices and transformed rank-1 lattices, including methods that use sampling sets consisting of Chebyshev- and tent-transformed nodes. We compare these methods with a parameterized transformed Fourier system that yields similar ℓ_2-approximation errors.

Keywords Change of variables · Lattice rule · Fast Fourier transform

1 Introduction

For the approximation of non-periodic functions defined on the cube $[0, 1]^d$, fast algorithms based on Chebyshev- and tent-transformed rank-1 lattice methods have been introduced and studied in [8, 10–12, 16, 19, 22]. Recently, we suggested a general framework for transformed rank-1 lattice approximation, in which functions defined on a cube $[0, 1]^d$ (or on $\mathbb{R}^d$) are periodized onto the torus $\mathbb{T}^d \simeq [0, 1)^d$, [17, 18]. In these approaches we define parameterized families $\psi(\cdot, \boldsymbol{\eta}) : [0, 1]^d \to [0, 1]^d$, $\boldsymbol{\eta} \in \mathbb{R}_+^d$ of transformations that, depending on the parameter choice, yield a certain smoothening effect when composed with a given non-periodic function. This periodization strategies also lead to general parameterized classes of orthonormal systems in weighted Hilbert spaces. However, these methods have the natural drawback of singularities appearing at the boundary points of the cube, so that any approximation error estimates have to be done with respect to weighted L_∞- and L_2-norms.

We summarize some crucial properties of rank-1 lattice approximation. Then, we compare the approximation with a half-periodic cosine system and tent-transformed

R. Nasdala (✉) · D. Potts
Faculty of Mathematics, Chemnitz University of Technology, D-09107 Chemnitz, Germany
e-mail: robert.nasdala@math.tu-chemnitz.de

D. Potts
e-mail: potts@math.tu-chemnitz.de

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_13

sampling nodes [1–3, 13, 22], the Chebyshev approximation [16, 19], as well as the general framework for the parameterized transformed Fourier system [18]. We discuss numerical results in up to dimension $d = 7$ and highlight the controlled smoothening effect when varying the parameter $\boldsymbol{\eta}$ in the transformed Fourier systems.

2 Approximation Methods

At first, we summarize the main ideas of the Fourier approximation with sampling sets in the form of rank-1 lattices [7, 14, 21]. Afterwards, we consider Chebyshev- and tent-transformed rank-1 lattices in the context of Chebyshev and cosine approximation methods [16, 22]. Finally, we outline the transformed Fourier system for the approximation of non-periodic signals, as introduced in [18], and provide two examples of parameterized transformations.

2.1 Fourier Approximation

For any frequency set $I \subset \mathbb{Z}^d$ of finite cardinality $|I| < \infty$ we denote the space of all multivariate trigonometric polynomials supported on I by

$$\Pi_I := \operatorname{span}\left\{ \mathrm{e}^{2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}} = \prod_{\ell=1}^{d} \mathrm{e}^{2\pi \mathrm{i} k_\ell x_\ell} : \mathbf{k} \in I, \mathbf{x} \in \mathbb{T}^d \right\}.$$

with $\mathbf{k} = (k_1, \ldots, k_d)^\top$, $\mathbf{x} = (x_1, \ldots, x_d)^\top$. The trigonometric polynomials are orthonormal with respect to the $L_2(\mathbb{T}^d)$-scalar product

$$(f, g)_{L_2(\mathbb{T}^d)} := \int_{\mathbb{T}^d} f(\mathbf{x})\, \overline{g(\mathbf{x})}\, \mathrm{d}\mathbf{x}, \quad f, g \in L_2(\mathbb{T}^d).$$

For all $\mathbf{k} \in \mathbb{Z}^d$ we denote the *Fourier coefficients* $\hat{h}_\mathbf{k}$ by

$$\hat{h}_\mathbf{k} := (h, \mathrm{e}^{2\pi \mathrm{i}\mathbf{k}(\cdot)})_{L_2(\mathbb{T}^d)} = \int_{\mathbb{T}^d} h(\mathbf{x})\, \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}}\, \mathrm{d}\mathbf{x},$$

and the corresponding *Fourier partial sum* by $S_I h(\mathbf{x}) := \sum_{\mathbf{k}\in I} \hat{h}_\mathbf{k}\, \mathrm{e}^{2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}}$.

We use sampling nodes in a *rank-1 lattice* $\Lambda(\mathbf{z}, M)$ of size $M \in \mathbb{N}$ generated by the vector $\mathbf{z} \in \mathbb{Z}^d$, that is defined as

$$\Lambda(\mathbf{z}, M) := \left\{ \mathbf{x}_j^{\text{latt}} := \frac{j}{M}\mathbf{z} \mod \mathbf{1} \in \mathbb{T}^d : j = 0, \ldots, M-1 \right\}, \tag{1}$$

with $\mathbf{1} := (1, \ldots, 1)^\top$, which allows the fast evaluation of Fourier partial sums via [14, Algorithm 3.1]. For any frequency set $I \subset \mathbb{Z}^d$ the *difference set* is given by

$$\mathcal{D}(I) := \{\mathbf{k} \in \mathbb{Z}^d : \mathbf{k} = \mathbf{k}_1 - \mathbf{k}_2 \text{ with } \mathbf{k}_1, \mathbf{k}_2 \in I\}. \tag{2}$$

We define the *reconstructing rank-1 lattice* $\Lambda(\mathbf{z}, M, I)$ as a rank-1 lattice $\Lambda(\mathbf{z}, M)$ for which the condition

$$\mathbf{t} \cdot \mathbf{z} \not\equiv 0 \,(\mathrm{mod}\, M) \quad \text{for all } \mathbf{t} \in \mathcal{D}(I) \setminus \{\mathbf{0}\} \tag{3}$$

holds.

Given a reconstructing rank-1 lattice $\Lambda(\mathbf{z}, M, I)$, we have exact integration for all multivariate trigonometric polynomials $p \in \Pi_{\mathcal{D}(I)}$, see [21], so that

$$\int_{\mathbb{T}^d} p(\mathbf{x})\,\mathrm{d}\mathbf{x} = \frac{1}{M} \sum_{j=0}^{M-1} p(\mathbf{x}_j), \quad \mathbf{x}_j \in \Lambda(\mathbf{z}, M, I). \tag{4}$$

In particular, for $h \in \Pi_I$ and $\mathbf{k} \in I$ we have $h(\cdot)\,\mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}(\cdot)} \in \Pi_{\mathcal{D}(I)}$ and

$$\hat{h}_\mathbf{k} = \int_{\mathbb{T}^d} h(\mathbf{x})\,\mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}}\,\mathrm{d}\mathbf{x} = \frac{1}{M} \sum_{j=0}^{M-1} h(\mathbf{x}_j)\,\mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}_j}, \quad \mathbf{x}_j \in \Lambda(\mathbf{z}, M, I). \tag{5}$$

Next, we focus on functions in the Wiener algebra $\mathcal{A}(\mathbb{T}^d)$ containing all $L_1(\mathbb{T}^d)$-functions with absolutely summable Fourier coefficients $\hat{h}_\mathbf{k}$ given by

$$\mathcal{A}(\mathbb{T}^d) := \left\{ h \in L_1(\mathbb{T}^d) : \sum_{k \in \mathbb{Z}^d} |\hat{h}_\mathbf{k}| < \infty \right\}. \tag{6}$$

For an arbitrary function $h \in \mathcal{A}(\mathbb{T}^d) \cap C(\mathbb{T}^d)$ and lattice points $\mathbf{x}_j \in \Lambda(\mathbf{z}, M, I)$ we lose the former mentioned exact integration property and get *approximated Fourier coefficients* $\hat{h}_\mathbf{k}^\Lambda$ of the form

$$\hat{h}_\mathbf{k} \approx \hat{h}_\mathbf{k}^\Lambda := \frac{1}{M} \sum_{j=0}^{M-1} h(\mathbf{x}_j)\,\mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}_j}$$

leading to the *approximated Fourier partial sum* $S_I^\Lambda h$ given by

$$h(\mathbf{x}) \approx S_I^{\Lambda} h(\mathbf{x}) := \sum_{\mathbf{k} \in I} \hat{h}_{\mathbf{k}}^{\Lambda} \, \mathrm{e}^{2\pi \mathrm{i} \mathbf{k} \cdot \mathbf{x}}.$$

For the matrix-vector-expression with respect to the frequency set $I_{\mathrm{latt}} \subset \mathbb{Z}^d$ we put

$$\mathbf{F}_{\mathrm{latt}} := \left\{ \mathrm{e}^{2\pi \mathrm{i} \mathbf{k} \cdot \mathbf{x}_j^{\mathrm{latt}}} \right\}_{j=0, \mathbf{k} \in I_{\mathrm{latt}}}^{M-1}, \quad \mathbf{h}_{\mathrm{latt}} := \left(h(\mathbf{x}_j^{\mathrm{latt}}) \right)_{j=0}^{M-1}.$$

The evaluation of the function h and the reconstruction of the approximated Fourier coefficients $\hat{\mathbf{h}} := (\hat{h}_{\mathbf{k}}^{\Lambda})_{\mathbf{k} \in I_{\mathrm{latt}}}$ are realized by the fast Algorithms outlined in [14, Algorithm 3.1 and 3.2] that solve the systems

$$\mathbf{h}_{\mathrm{latt}} = \mathbf{F}_{\mathrm{latt}} \hat{\mathbf{h}} \quad \text{and} \quad \hat{\mathbf{h}} = \frac{1}{M} \mathbf{F}_{\mathrm{latt}}^{*} \mathbf{h}_{\mathrm{latt}},$$

where we have $\mathbf{F}_{\mathrm{latt}}^{*} \mathbf{F}_{\mathrm{latt}} = M \mathbf{I}$ by construction with the identity matrix $\mathbf{I} \in \mathbb{C}^{|I_{\mathrm{latt}}| \times |I_{\mathrm{latt}}|}$.

2.2 Cosine Approximation

Next, we consider the half-periodic cosine system

$$\left\{ \lambda_{\mathbf{k}}(\mathbf{x}) := \sqrt{2}^{\|\mathbf{k}\|_0} \prod_{\ell=1}^{d} \cos(\pi k_\ell x_\ell) \right\}_{\mathbf{k} \in I_{\mathrm{tent}}}, \; I_{\mathrm{tent}} \subset \mathbb{N}_0^d, \mathbf{x} \in [0,1]^d, \tag{7}$$

with the zero-norm $\|\mathbf{k}\|_0 := |\{\ell \in \{1, \ldots, d\} : k_\ell \neq 0\}|$ and $\sqrt{2}^{\|\mathbf{k}\|_0} := \prod_{\ell=1}^{d} \sqrt{2}^{\|k_\ell\|_0}$. In [13] it is pointed out that this system can alternatively be defined in one dimension over the domain $t \in [-1, 1]$ as the system $\lambda_0(x) = \frac{1}{\sqrt{2}}$, $\lambda_k(t) = \cos(k\pi t)$, $\tilde{\lambda}_k(t) = \sin((k - \frac{1}{2})\pi t)$, which yields the original cosine system after applying the transformation $t = 2x - 1$.

The cosine system (7) is orthonormal with respect to the $L_2([0,1]^d)$-scalar product given by

$$(f, g)_{L_2([0,1]^d)} := \int_{[0,1]^d} f(\mathbf{x}) \, \overline{g(\mathbf{x})} \, \mathrm{d}\mathbf{x}, \quad f, g \in L_2([0,1]^d).$$

For $\mathbf{k} \in \mathbb{Z}^d$ the cosine coefficient of a function $h \in L_2\left([0,1]^d\right)$ is naturally defined as $\hat{h}_{\mathbf{k}}^{\cos} := (h, \lambda_{\mathbf{k}})_{L_2([0,1]^d)}$ and for $I \subset \mathbb{Z}^d$ the corresponding cosine partial sum is given by $S_I h(\mathbf{x}) := \sum_{\mathbf{k} \in I} \hat{h}_{\mathbf{k}}^{\cos} \lambda_{\mathbf{k}}(\mathbf{x})$. We transfer the crucial properties of the Fourier system via the tent transformation

$$\psi(\mathbf{x}) := (\psi_1(x_1), \ldots, \psi_d(x_d))^\top, \quad \psi_\ell(x_\ell) = \begin{cases} 2x_\ell & \text{for } 0 \le x_\ell < \frac{1}{2}, \\ 2 - 2x_\ell & \text{for } \frac{1}{2} \le x_\ell \le 1. \end{cases} \tag{8}$$

We have sampling nodes in the tent-transformed rank-1 lattice $\Lambda_\psi(\mathbf{z}, M)$ defined as

$$\Lambda_\psi(\mathbf{z}, M) := \left\{ \mathbf{y}_j^{\cos} := \psi\left(\mathbf{x}_j^{\text{latt}}\right) : \mathbf{x}_j^{\text{latt}} \in \Lambda(\mathbf{z}, M),\, j = 0, \ldots, M-1 \right\}$$

and we speak of a reconstructing tent-transformed rank-1 lattice $\Lambda_\psi(\mathbf{z}, M, I)$ if the underlying rank-1 lattice is a reconstructing one. Recalling the definition of difference sets $\mathcal{D}(I)$ in (2), multivariate trigonometric polynomials $h(\cdot), h(\cdot)$ and $\lambda_{\mathbf{k}}(\cdot)$ that are in $\Pi_{\mathcal{D}(I)}$ and supported on $\mathbf{k} \in I \subset \mathbb{N}_0^d$ inherit the exact integration property (4), because with the tent transformation as in (8) and transformed nodes $\mathbf{y}_j^{\cos} = \psi(\mathbf{x}_j^{\text{latt}}) \in \Lambda_\psi(\mathbf{z}, M, I)$ with $\mathbf{x}_j^{\text{latt}} = (x_1^j, \ldots, x_d^j)^\top \in \Lambda(\mathbf{z}, M, I)$ we have

$$\begin{aligned}
\hat{h}_{\mathbf{k}}^{\cos} = \int_{[0,1]^d} h(\mathbf{y})\, \lambda_{\mathbf{k}}(\mathbf{y})\, \mathrm{d}\mathbf{y} &= \sqrt{2}^{\|\mathbf{k}\|_0} \int_{\mathbb{T}^d} h(\psi(\mathbf{x})) \prod_{\ell=1}^{d} \cos(2\pi k_\ell x_\ell)\, \mathrm{d}\mathbf{x} \\
&= \frac{\sqrt{2}^{\|\mathbf{k}\|_0}}{2^d} \int_{\mathbb{T}^d} h(\psi(\mathbf{x})) \left(\mathrm{e}^{2\pi \mathrm{i} \mathbf{k}\cdot\mathbf{x}} + \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k}\cdot\mathbf{x}} \right) \mathrm{d}\mathbf{x} \\
&= \frac{\sqrt{2}^{\|\mathbf{k}\|_0}}{2^d} \frac{1}{M} \sum_{j=0}^{M-1} h(\psi(\mathbf{x}_j)) \left(\mathrm{e}^{2\pi \mathrm{i} \mathbf{k}\cdot\mathbf{x}_j} + \mathrm{e}^{-2\pi \mathrm{i} \mathbf{k}\cdot\mathbf{x}_j} \right) \\
&= \sqrt{2}^{\|\mathbf{k}\|_0} \frac{1}{M} \sum_{j=0}^{M-1} h(\psi(\mathbf{x}_j)) \prod_{\ell=1}^{d} \cos(2\pi k_\ell x_\ell^j) \\
&= \frac{1}{M} \sum_{j=0}^{M-1} h(\mathbf{y}_j^{\cos})\, \lambda_{\mathbf{k}}(\mathbf{y}_j^{\cos}).
\end{aligned}$$

For an arbitrary function $h \in C\left([0,1]^d\right)$, we lose the former mentioned exactness and define the *approximated cosine coefficients* $\hat{h}_{\mathbf{k}}^{\cos,\Lambda}$ of the form

$$\hat{h}_{\mathbf{k}}^{\cos} \approx \hat{h}_{\mathbf{k}}^{\cos,\Lambda} := \frac{1}{M} \sum_{j=0}^{M-1} h(\mathbf{y}_j^{\cos})\, \lambda_{\mathbf{k}}(\mathbf{y}_j^{\cos}), \quad \mathbf{y}_j^{\cos} \in \Lambda_\psi(\mathbf{z}, M, I),$$

and obtain *approximated cosine partial sum* $S_I^\Lambda h$ given by

$$h(\mathbf{x}) \approx S_I^\Lambda h(\mathbf{x}) := \sum_{\mathbf{k} \in I} \hat{h}_{\mathbf{k}}^{\cos,\Lambda}\, \lambda_{\mathbf{k}}(\mathbf{x}). \tag{9}$$

In matrix-vector-notation we have

$$\mathbf{C} := \left\{\lambda_{\mathbf{k}}\left(\mathbf{y}_j^{\cos}\right)\right\}_{j=0,\mathbf{k}\in I_{\text{tent}}}^{M-1}, \quad \mathbf{h}_{\text{tent}} := \left(h(\mathbf{y}_j^{\cos})\right)_{j=0}^{M-1}.$$

Both the evaluation of h and the reconstruction of the approximated cosine coefficients $\hat{\mathbf{h}} := \left\{\hat{h}_{\mathbf{k}}^{\cos,\Lambda}\right\}_{\mathbf{k}\in I_{\text{tent}}}$ is realized by solving the systems

$$\mathbf{h}_{\text{tent}} = \mathbf{C}\hat{\mathbf{h}} \quad \text{and} \quad \hat{\mathbf{h}} = \frac{1}{M}\mathbf{C}^*\mathbf{h}_{\text{tent}}, \tag{10}$$

where we have $\mathbf{C}^*\mathbf{C} = M\mathbf{I}$ by construction with the identity matrix $\mathbf{I} \in \mathbb{C}^{|I_{\text{tent}}|\times|I_{\text{tent}}|}$. Fast algorithms for solving both systems are described in [16, 22].

2.3 Chebyshev Approximation

We consider the Chebyshev system, that is defined for $\mathbf{x} \in [0,1]^d$ and a finite frequency set $I_{\text{cheb}} \subset \mathbb{N}_0^d$ as

$$\left\{T_{\mathbf{k}}(\mathbf{x}) := \sqrt{2}^{\|\mathbf{k}\|_0} \prod_{\ell=1}^{d} \cos\left(k_\ell \arccos(2x_\ell - 1)\right)\right\}_{\mathbf{k}\in I_{\text{cheb}}}. \tag{11}$$

The Chebyshev system (11) is an orthonormal system with respect to the weighted scalar product

$$(T_{\mathbf{k}_1}, T_{\mathbf{k}_2})_{L_2([0,1]^d,\omega)} := \int_{[0,1]^d} T_{\mathbf{k}_1}(\mathbf{x})\, T_{\mathbf{k}_2}(\mathbf{x})\, \omega(\mathbf{x})\, \mathrm{d}\mathbf{x}, \ \omega(\mathbf{x}) := \prod_{\ell=1}^{d} \frac{2}{\pi\sqrt{4x_\ell(1-x_\ell)}}.$$

The *Chebyshev coefficients* of a function $h \in L_2\left([0,1]^d, \omega\right)$ are naturally defined as $\hat{h}_{\mathbf{k}}^{\text{cheb}} := (h, T_{\mathbf{k}})_{L_2([0,1]^d,\omega)}$, $\mathbf{k} \in \mathbb{Z}^d$ and for $I \subset \mathbb{Z}^d$ the corresponding Chebyshev partial sum is given by $S_I h(\mathbf{x}) := \sum_{\mathbf{k}\in I} \hat{h}_{\mathbf{k}}^{\text{cheb}}\, T_{\mathbf{k}}(\mathbf{x})$. We transfer some properties of the Fourier system via the *Chebyshev transformation*

$$\begin{aligned}
\psi(\mathbf{x}) &:= (\psi_1(x_1), \ldots, \psi_d(x_d))^\top, \\
\psi_\ell(x_\ell) &:= \frac{1}{2} + \frac{1}{2}\cos\left(2\pi\left(x_\ell - \frac{1}{2}\right)\right), \quad x_\ell \in [0,1], \quad \ell \in \{1,\ldots,d\}.
\end{aligned} \tag{12}$$

We have sampling nodes in the Chebyshev-transformed rank-1 lattice $\Lambda_\psi(\mathbf{z}, M)$ defined as

$$\Lambda_\psi(\mathbf{z}, M) := \left\{\mathbf{y}_j^{\text{cheb}} := \psi\left(\mathbf{x}_j^{\text{latt}}\right) : \mathbf{x}_j^{\text{latt}} \in \Lambda(\mathbf{z}, M),\, j = 0, \ldots, M-1\right\}.$$

It inherits the reconstruction property (3) of the underlying reconstructing rank-1 lattice $\Lambda(\mathbf{z}, M, I)$ and is denoted by $\Lambda_\psi(\mathbf{z}, M, I)$. We note that Chebyshev transformed sampling nodes are fundamentally connected to Padua points and Lissajous curves, as well as certain interpolation methods that are outlined in [4, 9].

Recalling the definition of difference sets $\mathcal{D}(I)$ in (2), multivariate trigonometric polynomials $h(\cdot)$ and $h(\cdot)\, T_{\mathbf{k}}(\cdot)$ are in $\Pi_{\mathcal{D}(I)}$ and supported on $\mathbf{k} \in I \subset \mathbb{N}_0^d$ inherit the exact integration property (4), because with the Chebyshev transformation ψ as in (12) and transformed nodes $\mathbf{y}_j^{\text{cheb}} = \psi(\mathbf{x}_j^{\text{latt}}) \in \Lambda_\psi(\mathbf{z}, M, I)$ with $\mathbf{x}_j^{\text{latt}} = (x_1^j, \ldots, x_d^j)^\top \in \Lambda(\mathbf{z}, M, I)$ we have

$$\begin{aligned}
\hat{h}_{\mathbf{k}}^{\text{cheb}} = \int_{[0,1]^d} h(\mathbf{y})\, T_{\mathbf{k}}(\mathbf{y})\, \omega(\mathbf{y})\, \mathrm{d}\mathbf{y} &= \sqrt{2}^{\|\mathbf{k}\|_0} \int_{\mathbb{T}^d} h(\psi(\mathbf{x})) \prod_{\ell=1}^{d} \cos(2\pi k_\ell x_\ell)\, \mathrm{d}\mathbf{x} \\
&= \frac{\sqrt{2}^{\|\mathbf{k}\|_0}}{2^d} \int_{\mathbb{T}^d} h(\psi(\mathbf{x})) \left(\mathrm{e}^{2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}} + \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}}\right) \mathrm{d}\mathbf{x} \\
&= \frac{\sqrt{2}^{\|\mathbf{k}\|_0}}{2^d} \frac{1}{M} \sum_{j=0}^{M-1} h(\psi(\mathbf{x}_j)) \left(\mathrm{e}^{2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}_j} + \mathrm{e}^{-2\pi \mathrm{i}\mathbf{k}\cdot\mathbf{x}_j}\right) \\
&= \sqrt{2}^{\|\mathbf{k}\|_0} \frac{1}{M} \sum_{j=0}^{M-1} h(\psi(\mathbf{x}_j)) \prod_{\ell=1}^{d} \cos(2\pi k_\ell x_\ell^j) \\
&= \frac{1}{M} \sum_{j=0}^{M-1} h(\mathbf{y}_j^{\text{cheb}})\, T_{\mathbf{k}}(\mathbf{y}_j^{\text{cheb}}).
\end{aligned}$$

For an arbitrary function $h \in L\left([0,1]^d, \omega\right) \cap C\left([0,1]^d\right)$, we lose the former mentioned exactness and define the *approximated Chebyshev coefficients* $\hat{h}_{\mathbf{k}}^{\text{cheb},\Lambda}$ of the form

$$\hat{h}_{\mathbf{k}}^{\text{cheb}} \approx \hat{h}_{\mathbf{k}}^{\text{cheb},\Lambda} := \frac{1}{M} \sum_{j=0}^{M-1} h(\mathbf{y}_j^{\text{cheb}})\, T_{\mathbf{k}}(\mathbf{y}_j^{\text{cheb}}), \quad \mathbf{y}_j^{\text{cheb}} \in \Lambda_\psi(\mathbf{z}, M, I),$$

leading to the *approximated Chebyshev partial sum*

$$h(\mathbf{x}) \approx S_I^\Lambda h(\mathbf{x}) := \sum_{\mathbf{k}\in I} \hat{h}_{\mathbf{k}}^{\text{cheb},\Lambda}\, T_{\mathbf{k}}(\mathbf{x}). \tag{13}$$

In matrix-vector-notation this reads as

$$\mathbf{T} := \left\{T_{\mathbf{k}}(\mathbf{y}_j^{\text{cheb}})\right\}_{j=0,\mathbf{k}\in I_{\text{cheb}}}^{M-1}, \quad \mathbf{h}_{\text{cheb}} := \left(h(\mathbf{y}_j^{\text{cheb}})\right)_{j=0}^{M-1}.$$

The evaluation of h as well as the reconstruction of the approximated Chebyshev coefficients $\hat{\mathbf{h}} := \left(\hat{h}_{\mathbf{k}}^{\mathrm{cheb},\Lambda}\right)_{\mathbf{k}\in I_{\mathrm{cheb}}}$ of h are realized by fast Algorithms outlined in [16, 19, 22], that solve the systems

$$\mathbf{h}_{\mathrm{cheb}} = \mathbf{T}\hat{\mathbf{h}} \quad \text{and} \quad \hat{\mathbf{h}} = \frac{1}{M}\mathbf{T}^{*}\mathbf{h}_{\mathrm{cheb}}, \tag{14}$$

where we have $\mathbf{T}^{*}\mathbf{T} = M\mathbf{I}$ by construction with the identity matrix $\mathbf{I} \in \mathbb{C}^{|I_{\mathrm{cheb}}|\times|I_{\mathrm{cheb}}|}$.

2.4 Transformed Fourier Approximation

We recall the ideas of a particular family of parameterized torus-to-cube transformations as suggested in [18], that generalize the construction idea of the Chebyshev system in composing a mapping with a multiple of its inverse.

We call a continuously differentiable, strictly increasing mapping $\tilde{\psi} : (0, 1) \to \mathbb{R}$ with $\tilde{\psi}(x + \frac{1}{2})$ being odd and $\tilde{\psi}(x) \to \pm\infty$ for $x \to \{0, 1\}$ a *torus-to-*$\mathbb{R}$ *transformation*. We obtain a parameterized *torus-to-cube transformation* $\psi(\cdot, \eta) : [0, 1] \to [0, 1]$ with $\eta \in \mathbb{R}_{+} := (0, \infty)$ by putting

$$\psi(x, \eta) := \begin{cases} 0 & \text{for} \quad x = 0, \\ \tilde{\psi}^{-1}(\eta\, \tilde{\psi}(x)) & \text{for} \quad x \in (0, 1)\,, \\ 1 & \text{for} \quad x = 1, \end{cases} \tag{15}$$

which are continuously differentiable, increasing and have a first derivative $\psi'(\cdot, \eta) \in C_0([0, 1])$, where $C_0([0, 1])$ denotes the space of all continuous functions vanishing to 0 towards their boundary points. It holds $\psi^{-1}(y, \eta) = \psi\left(y, \frac{1}{\eta}\right)$ and we call $\varrho(y, \eta) := (\psi^{-1})'(y, \eta) = \psi'\left(y, \frac{1}{\eta}\right)$ the *density of* ψ. In multiple dimensions $d \in \mathbb{N}$ with $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_d)^{\top}$ we put

$$\begin{aligned} \psi(\mathbf{x}, \boldsymbol{\eta}) &:= (\psi_1(x_1, \eta_1), \ldots, \psi_d(x_d, \eta_d))^{\top}, \\ \psi^{-1}(\mathbf{y}, \boldsymbol{\eta}) &:= (\psi_1^{-1}(y_1, \eta_1), \ldots, \psi_d^{-1}(y_d, \eta_d))^{\top}, \\ \varrho(\mathbf{y}, \boldsymbol{\eta}) &:= \prod_{\ell=1}^{d} \varrho_\ell(y_\ell, \eta_\ell) \quad \text{with} \quad \varrho_\ell(y_\ell, \eta_\ell) := \frac{1}{\psi'(\psi^{-1}(y_\ell, \eta_\ell))}, \end{aligned} \tag{16}$$

where the univariate torus-to-cube transformations $\psi_\ell(\cdot, \eta_\ell)$ and their corresponding densities $\varrho_\ell(\cdot, \eta_\ell)$ may be different in each coordinate $\ell \in \{1, \ldots, d\}$.

We consider integrable weight functions

$$\omega(\mathbf{y}) := \prod_{\ell=1}^{d} \omega_\ell(y_\ell), \quad \mathbf{y} \in [0,1]^d,$$

such that for any given torus-to-cube transformation $\psi(\cdot, \boldsymbol{\eta})$ as in (16) we have

$$\omega(\psi_\ell(\cdot, \eta_\ell))\psi'(\cdot, \eta_\ell) \in C_0([0,1]).$$

Applying a torus-to-cube transformation to a function $h \in L_2([0,1]^d, \omega) \cap C([0,1]^d)$ generates a periodic function $f \in L_2(\mathbb{T}^d)$ of the form

$$f(\mathbf{x}) := h(\psi(\mathbf{x}, \boldsymbol{\eta})) \sqrt{\omega(\psi(\mathbf{x}, \boldsymbol{\eta})) \prod_{\ell=1}^{d} \psi'_\ell(x_\ell)} \quad \text{with} \quad \|h\|_{L_2([0,1]^d,\omega)} = \|f\|_{L_2(\mathbb{T}^d)}, \tag{17}$$

that is approximated by the classical Fourier system. To construct an approximant for the original function h we apply the inverse torus-to-cube transformation to the Fourier system, yielding for a fixed $\boldsymbol{\eta} \in \mathbb{R}^d_+$ the *transformed Fourier system*

$$\left\{ \varphi_{\mathbf{k}}(\cdot) := \sqrt{\frac{\varrho(\cdot, \boldsymbol{\eta})}{\omega(\cdot)}}\, \mathrm{e}^{2\pi \mathrm{i}\mathbf{k}\cdot\psi^{-1}(\cdot,\boldsymbol{\eta})} \right\}_{\mathbf{k}\in I}, \tag{18}$$

which forms an orthonormal system with respect to the weighted $L_2\left([0,1]^d, \omega\right)$-scalar product. For all $\mathbf{k} \in \mathbb{Z}^d$ the *transformed Fourier coefficients* $\hat{h}_{\mathbf{k}}$ are naturally defined as

$$\hat{h}_{\mathbf{k}} := (h, \varphi_{\mathbf{k}})_{L_2([0,1]^d,\omega)} = \int_{[0,1]^d} h(\mathbf{y})\, \overline{\varphi_{\mathbf{k}}(\mathbf{y})}\, \omega(\mathbf{y})\, \mathrm{d}\mathbf{y},$$

and the corresponding *Fourier partial sum* is given by $S_I h(\mathbf{y}) := \sum_{\mathbf{k}\in I} \hat{h}_{\mathbf{k}}\, \varphi(\mathbf{y})$. The corresponding sampling nodes will be taken from the torus-to-cube-transformed (abbreviated: ttc) rank-1 lattice $\Lambda_\psi(\mathbf{z}, M)$ defined as

$$\Lambda_\psi(\mathbf{z}, M) := \left\{ \mathbf{y}_j^{\mathrm{ttc}} := \psi\left(\mathbf{x}_j^{\mathrm{latt}}, \boldsymbol{\eta}\right) : \mathbf{x}_j^{\mathrm{latt}} \in \Lambda(\mathbf{z}, M),\, j = 0, \ldots, M-1 \right\}$$

and we speak of a reconstructing torus-to-cube-transformed rank-1 lattice $\Lambda_\psi(\mathbf{z}, M, I)$ if the underlying rank-1 lattice is a reconstructing one.

Furthermore, the multivariate transformed trigonometric polynomials supported on $I \subset \mathbb{Z}^d$ are given by $\Pi_I^{\mathrm{ttc}} := \mathrm{span}\{\varphi_{\mathbf{k}} : \mathbf{k} \in I\}$ and inherit the exact integration property (5), thus, for $h \in \Pi_I^{\mathrm{ttc}}$ we have

$$\hat{h}_{\mathbf{k}} = \int_{[0,1]^d} h(\mathbf{x})\, \varphi_{\mathbf{k}}(\mathbf{x})\, \mathrm{d}\mathbf{x} = \frac{1}{M} \sum_{j=0}^{M-1} h(\mathbf{y}_j^{\mathrm{ttc}})\, \varphi_{\mathbf{k}}(\mathbf{y}_j^{\mathrm{ttc}}), \quad \mathbf{y}_j^{\mathrm{ttc}} \in \Lambda_\psi(\mathbf{z}, M, I).$$

For an arbitrary function $h \in L_2([0,1]^d, \omega) \cap C([0,1]^d)$ we lose the former mentioned exactness and define approximated transformed coefficients of the form

$$\hat{h}_{\mathbf{k}}^{\Lambda} := \frac{1}{M} \sum_{j=0}^{M-1} h(\mathbf{y}_j^{\mathrm{ttc}})\, \varphi_{\mathbf{k}}(\mathbf{y}_j^{\mathrm{ttc}})$$

and leads to the *approximated transformed Fourier partial sum* $S_I^{\Lambda} h$ given by

$$h(\mathbf{y}) \approx S_I^{\Lambda} h(\mathbf{y}) := \sum_{\mathbf{k} \in I} \hat{h}_{\mathbf{k}}^{\Lambda}\, \varphi_{\mathbf{k}}(\mathbf{y}). \tag{19}$$

In matrix-vector-notation we have

$$\mathbf{h}_{\mathrm{ttc}} := \left(h(\mathbf{y}_j^{\mathrm{ttc}})\right)_{j=0}^{M-1}, \quad \mathbf{F}_{\mathrm{ttc}} := \left\{\varphi_{\mathbf{k}}\left(\mathbf{y}_j^{\mathrm{ttc}}\right)\right\}_{j=0,\mathbf{k}\in I_{\mathrm{ttc}}}^{M-1}.$$

The evaluation of h and the reconstruction of the approximated transformed Fourier coefficients $\hat{\mathbf{h}} := \left\{\hat{h}_{\mathbf{k}}^{\Lambda}\right\}_{\mathbf{k}\in I_{\mathrm{ttc}}}$ is realized by solving the systems

$$\mathbf{h}_{\mathrm{ttc}} = \mathbf{F}_{\mathrm{ttc}} \hat{\mathbf{h}}. \quad \text{and} \quad \hat{\mathbf{h}} = \frac{1}{M} \mathbf{F}_{\mathrm{ttc}}^{*} \mathbf{h}_{\mathrm{ttc}}. \tag{20}$$

Fast algorithms for solving both systems are described in [18].

2.5 Comparison of the Orthonormal Systems

The previously presented approximation approaches are based on very different orthonormal systems and use differently transformed sampling sets, which is summarized in dimension $d = 1$ in Table 1 with the definition of the hyperbolic cross I_N^1 given in (23).

Given an univariate continuous function $h \in C([0,1])$, both composition with the tent transformation (8) and the Chebyshev transformation (12) can be interpreted as mirroring a compressed version of h at the point $\frac{1}{2}$, so that $h(\psi(x)) = h(\psi(1-x))$ for all $x \in [0, \frac{1}{2}]$. In contrast to the the Chebyshev transformation case, for the tent transformation we generally won't expect the resulting function $h \circ \psi$ to be smooth at the point $\frac{1}{2}$, which will be reflected in the approximation results later on.

The parametrized torus-to-cube transformations (15) are a fundamentally different transformation class in the sense that the periodization effect is caused primarily by

Table 1 Comparison of the univariate orthonormal system, sampling sets and frequency sets from the Chebyshev, cosine and transformed Fourier approximation methods

Orthonormal system $\{\varphi_k(x)\}_{k\in I}$	Scalar product weight ω	Sampling transformation ψ	Frequency set I
$\sqrt{2}^{\|k\|_0}\cos(\pi kx)$	1	$\begin{cases} 2x & \text{for } 0 \le x < \frac{1}{2}, \\ 2-2x & \text{for } \frac{1}{2} \le x \le 1. \end{cases}$	$I_N^d \cap \mathbb{N}_0^d$
$\sqrt{2}^{\|k\|_0}\cos(k\arccos(2x-1))$	$\frac{1}{2\pi\sqrt{x(1-x)}}$	$\frac{1}{2}+\frac{1}{2}\cos\left(2\pi(x-\frac{1}{2})\right)$	$I_N^d \cap \mathbb{N}_0^d$
$\sqrt{\frac{\varrho(x,\eta)}{\omega(x)}}\,\mathrm{e}^{2\pi \mathrm{i} k\psi^{-1}(x,\eta)}$	$\omega(x)$	$\psi(x,\eta)$	I_N^d

the multiplication of $h(\psi(\cdot,\eta))$ with the first derivative $\psi(\cdot,\eta) \in C_0([0,1])$ (assuming a constant weight function $\omega \equiv 1$), so that the function $h(\psi(\cdot,\eta))\sqrt{\psi(\cdot,\eta)}$ ends up being continuously extendable to the torus $\mathbb{T}$. Additionally, now there is the parameter η involved which controls the smoothening effect on the periodized function, see [18].

We find various suggestions for torus-to-$\mathbb{R}$ transformations in [5, Sect. 17.6], [20, Sect. 7.5] and [17]. We list some induced combined transformations $\psi(x,\eta)$ and the corresponding density function $\varrho(y,\eta) = (\psi^{-1})'(y,\eta)$ in the sense of definition (15):

- the logarithmic torus-to-cube transformation

$$\psi(x,\eta) := \frac{1}{2} + \frac{1}{2}\tanh(\eta\tanh^{-1}(2x-1)), \quad \varrho(y,\eta) = \frac{4}{\eta}\frac{(4y-4y^2)^{\frac{1}{\eta}-1}}{\left((2y)^{\frac{1}{\eta}} + (2-2y)^{\frac{1}{\eta}}\right)^2}, \tag{21}$$

  based on the mapping

$$\tilde{\psi}(x) = \frac{1}{2}\log\left(\frac{2x}{2-2x}\right) = \tanh^{-1}(2x-1),$$

- the error function torus-to-cube transformation

$$\psi(x,\eta) = \frac{1}{2}\operatorname{erf}(\eta\operatorname{erf}^{-1}(2x-1)) + \frac{1}{2}, \quad \varrho(y,\eta) = \frac{1}{\eta}\,\mathrm{e}^{(1-\frac{1}{\eta^2})(\operatorname{erf}^{-1}(2y-1))^2}, \tag{22}$$

  based on the mapping

$$\tilde{\psi}(x) = \operatorname{erf}^{-1}(2x-1),$$

  which is the inverse of the error function

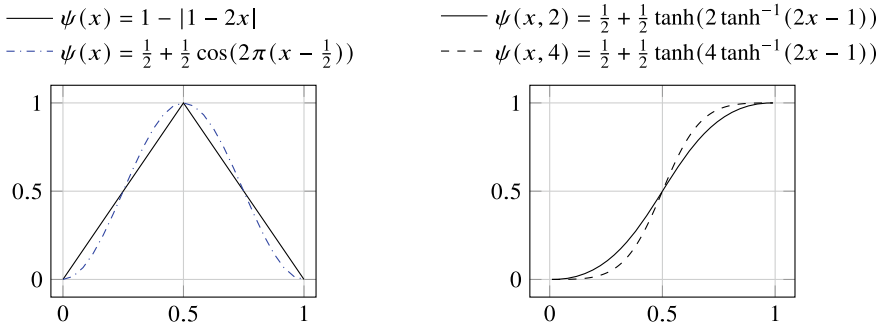


Fig. 1 Left: The tent-transformation (8) the Chebyshev-transformation (12). Right: The parameterized logarithmic transformation (21) for $\eta \in \{2, 4\}$

$$\operatorname{erf}(y) = \frac{1}{\sqrt{\pi}} \int_{-y}^{y} \mathrm{e}^{-t^2} \, \mathrm{d}t, \quad y \in \mathbb{R},$$

In Fig. 1 we provide a side-by-side comparison of all the previously mentioned transformation mappings.

3 Approximation Results and Error Analysis

Based on the weight function

$$\omega_{\mathrm{hc}}(\mathbf{k}) := \prod_{\ell=1}^{d} \max(1, |k_\ell|), \quad \mathbf{k} \in \mathbb{Z}^d,$$

we define the hyperbolic cross index set

$$I_N^d := \left\{ \mathbf{k} \in \mathbb{Z}^d : \omega_{\mathrm{hc}}(\mathbf{k}) \leq N \right\} \tag{23}$$

and for $\beta \geq 0$ we furthermore have the Hilbert spaces

$$\mathcal{H}^\beta(\mathbb{T}^d) := \left\{ f \in L_2(\mathbb{T}^d) : \|f\|_{\mathcal{H}^\beta(\mathbb{T}^d)}^2 := \sum_{\mathbf{k} \in \mathbb{Z}^d} \omega_{\mathrm{hc}}(\mathbf{k})^{2\beta} |\hat{f}_{\mathbf{k}}|^2 < \infty \right\} \tag{24}$$

that are closely related to the Wiener Algebra $\mathcal{A}(\mathbb{T}^d)$ given in (6). For $\lambda > \frac{1}{2}$ and fixed $d \in \mathbb{N}$ the continuous embeddings $\mathcal{H}^{\beta+\lambda}(\mathbb{T}^d) \hookrightarrow \mathcal{A}(\mathbb{T}^d)$ was shown in [15, Lemma 2.2]. Next, we introduce the analogue on the cube $[0, 1]^d$ for the Hilbert

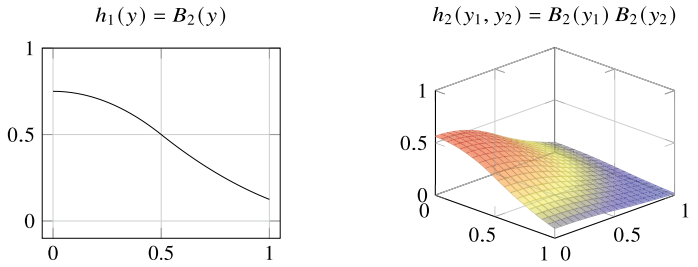


Fig. 2 The univariate B_2-cutoff $h_1(y) = B_2(y)$ and the two-dimensional tensored B_2-cutoff $h_2(y_1, y_2) = B_2(y_1)\, B_2(y_2)$

space $\mathcal{H}^\beta(\mathbb{T}^d)$ as in (24). We define the space of weighted $L_2\left([0,1]^d, \omega\right)$-functions with square summable Fourier coefficients $\hat{h}_{\mathbf{k}} := (h, \varphi_{\mathbf{k}})_{L_2([0,1]^d,\omega)}$ by

$$\mathcal{H}^\beta\left([0,1]^d, \omega\right) := \left\{h \in L_2\left([0,1]^d, \omega\right) : \|h\|_{\mathcal{H}^\beta([0,1]^d,\omega)} < \infty\right\},$$
$$\|h\|^2_{\mathcal{H}^\beta([0,1]^d,\omega)} := \sum_{\mathbf{k}\in\mathbb{Z}^d} \omega_{\mathrm{hc}}(\mathbf{k})^{2\beta} |\hat{h}_{\mathbf{k}}|^2. \tag{25}$$

In case of a constant weight function $\omega \equiv 1$ we just write $\mathcal{H}^\beta\left([0,1]^d\right)$.

We define a shifted, scaled and dilated B-spline of second order as

$$B_2(x) := \begin{cases} -x^2 + \frac{3}{4} & \text{for } 0 \le x < \frac{1}{2}, \\ \frac{1}{2}\left(x^2 - 3x + \frac{9}{4}\right) & \text{for } \frac{1}{2} \le x \le 1, \end{cases} \tag{26}$$

which we refer to as the B_2 *-cutoff*, that was also used in [18, 19]. It is in $C^1([0,1])$ and depicted in Fig. 2. Even though it is only once continuously differentiable, it is also an element in $\mathcal{H}^{\frac{5}{2}-\varepsilon}([0,1])$ for any $\varepsilon > 0$, which the following arguments show. It's well-known a second order B-spline is the result of a convolution of three step functions $\chi_{[0,1]}$ (where χ denotes the indicator function) with themselves, whose respective Fourier coefficients $(\chi_{[0,1]}(\cdot), \mathrm{e}^{2\pi i k(\cdot)})_{L_2([0,1])}$ decay like $|k|^{-1}$ for $k \to \pm\infty$. Hence, the Fourier coefficients $\hat{h}_k = (B_2, \mathrm{e}^{2\pi i k(\cdot)})_{L_2([0,1])}$ of the B_2-cutoff (26) decay like $|k|^{-3}$ for $k \to \pm\infty$. Considering a constant weight function $\omega \equiv 1$, the $\|\cdot\|_{\mathcal{H}^\beta([0,1])}$-norm given in (25) of B_2 is finite if

$$\|B_2\|^2_{\mathcal{H}^\beta([0,1])} = \sum_{k\in\mathbb{Z}} \omega_{\mathrm{hc}}(k)^{2\beta} |\hat{h}_k|^2 \lesssim \sum_{k\in\mathbb{Z}} \max\{1, |k|\}^{2\beta} \frac{1}{|k|^6} < \infty,$$

which is the case for

$$|k|^{2\beta-6} \le k^{-(1+\varepsilon)} \Leftrightarrow \beta \le \frac{5}{2} - \varepsilon, \quad \varepsilon > 0.$$

Next, we approximate the tensored B_2-cutoff

$$h(\mathbf{x}) = \prod_{\ell=1}^{d} B_2(x_\ell) \in \mathcal{H}^{\frac{5}{2}-\varepsilon}([0,1]^d), \quad \varepsilon > 0, \tag{27}$$

by the approximated Chebyshev, cosine or transformed Fourier partial sums $S_I^\Lambda h$ given in (9), (13) and (19). We study the resulting relative ℓ_2-and ℓ_∞-approximation errors

$$\varepsilon_p^R(h) := \frac{\left\| \left(h(\mathbf{x}_j) - S_I^\Lambda h(\mathbf{x}_j) \right)_{j=1}^R \right\|_{\ell_p}}{\left\| \left(h(\mathbf{x}_j) \right)_{j=1}^R \right\|_{\ell_p}}, \quad p \in \{2, \infty\}, \tag{28}$$

that are evaluated at $R \in \mathbb{N}$ uniformly distributed points $\mathbf{x}_j \sim \mathcal{U}([0,1]^d)$. The approximated coefficients appearing in the approximated partial sums (13), (9) and (19) are calculated by solving the corresponding systems (14), (10) or (20).

3.1 The Numerical Results of ℓ_2-Approximation

Throughout this section we repeatedly use the bold number notation $\mathbf{1} = (1, \ldots, 1)^\top$ that we already used in the definition of rank-1 lattices (1) and expressions like $\boldsymbol{\eta} = \mathbf{2}$ mean that $\eta_\ell = 2$ for all $\ell \in \{1, \ldots, d\}$.

In [6, 23, 24] we find a broad discussion on the approximation error decay of function in the Sobolev space $\mathcal{H}^\beta(\mathbb{T}^d)$, $m \in \mathbb{N}_0$. It was proven that there is a worst case upper error bound of the form

$$\varepsilon_2^R(h) \approx \left\| h - S_{I_N^d}^\Lambda h \right\|_{L_2([0,1]^d)} \lesssim N^{-m} (\log N)^{(d-1)/2}. \tag{29}$$

In [18] we find conditions on the logarithmic and the error function transformation $\psi(\cdot, \boldsymbol{\eta})$, given in (21) and (22), such that a certain degree of smoothness of the given $C^m(\mathbb{T}^d)$-function is preserved under composition with $\psi(\cdot, \boldsymbol{\eta})$ and the resulting periodized function is at least in $\mathcal{H}^\ell(\mathbb{T}^d)$, $\ell \leq m$ and for each ℓ it was calculated how large the parameter η has to be chosen. According to the conditions in [18, Theorem 4], the tensored B_2-cutoff in (27) is transformed into a function $f \in \mathcal{H}^0(\mathbb{T}^d)$ of the form (17) for all considered torus-to-cube transformations $\psi(\cdot, \boldsymbol{\eta})$ with parameters $1 < \eta_\ell \leq 3$, and into a function $f \in \mathcal{H}^1(\mathbb{T}^d)$ for parameters $\eta_\ell > 3$, $\ell \in \{1, \ldots, d\}$. While these conditions are independent of the particular considered function $h \in C^m(\mathbb{T}^d)$, they are pretty coarse in the sense of not catching the additional smoothness of functions like the B_2-cutoff given in (27) which is an almost $\mathcal{H}^{\frac{5}{2}}([0,1]^d)$-function as we showed earlier. In numerical tests we showcase that in certain setups the Chebyshev

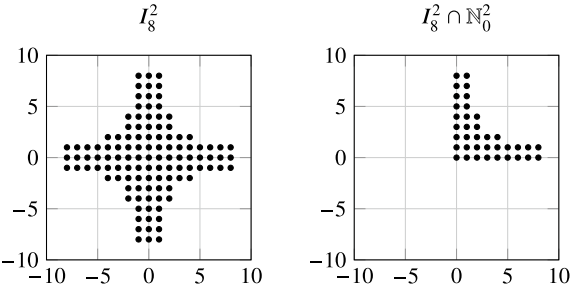


Fig. 3 The hyperbolic cross I_8^2 (left) and its first quadrant $I_8^2 \cap \mathbb{N}_0^2$ (right)

coefficients and the transformed Fourier coefficients will indeed decay faster than the worst case upper bound (29).

In dimensions $d \in \{1, 2, 4, 7\}$ we compare the discrete ℓ_2-approximation error ε_2, given in (28), with $R = 1.000.000$ uniformly distributed evaluation points for all of the previously introduced approximation approaches. We consider frequency sets I_N^d for all transformed Fourier systems and $I_N^d \cap \mathbb{N}_0^d$ for the cosine and Chebyshev systems. Both frequency sets are illustrated in dimension $d = 2$ with $N = 8$ in Fig. 3.

We use $N \in \{1, \ldots, 140\}$ for $d = 1$, $N \in \{1, \ldots, 80\}$ for $d = 2$, $N \in \{1, \ldots, 50\}$ for $d = 4$ and $N \in \{1, \ldots, 30\}$ for $d = 7$.

In dimensions $d = 1$ and $d = 2$ we observe that the approximation errors are significantly better for $\boldsymbol{\eta} = \mathbf{4}$ than for $\boldsymbol{\eta} = \mathbf{2}$, indicating the increased smoothening effect of both the logarithmic and the error function transformation. In dimensions $d \in \{4, 7\}$, the errors for $\boldsymbol{\eta} = \mathbf{4}$ turn out to be worse than for $\boldsymbol{\eta} = \mathbf{2}$, which we suspect might be due to the increase of certain constants depending on $\boldsymbol{\eta}$ in the error estimate (29). The Chebyshev approximation turns out to be a solid candidate to approximate the B-spline given in (27). In this specific setup, we also checked the error behavior for other parameters $\boldsymbol{\eta} \in \{\mathbf{2.1}, \mathbf{2.2}, \ldots, \mathbf{3.8}, \mathbf{3.9}, \mathbf{4.1}, \mathbf{4.2}, \ldots\}$. As it turns out, $\boldsymbol{\eta} = \mathbf{4}$ is the best choice for the logarithmic transformation and for the error function transformation the best choice is $\boldsymbol{\eta} = \mathbf{2.5}$.

However, as shown in Fig. 4, only the error function transformation is able to match the approximation error of the Chebyshev approximation, which also shows when we investigate and compare the error decay rates of $\varepsilon_2^R(h)$ that were numerically observed for the univariate case $d = 1$. In this specific setup, h is still the continuous second-order B-spline given in (27) that is an element of $\mathcal{H}^{\frac{5}{2}-\varepsilon}\left([0,1]^d\right)$. Hence, we expect to obtain an error decay at most $\varepsilon_2^R(h) \lesssim N^{-\frac{5}{2}+\varepsilon}$ for any $\varepsilon > 0$ and increasing values of N when approximating h with respect to any transformed Fourier system. We achieve these decay rates numerically with the Chebyshev system and with the transformed Fourier system when considering the logarithmic transformation with $\eta \in \{2.5, 4\}$. Interestingly, the decay rates of the cosine system remain at $N^{-1.5}$. In comparison, the logarithmically transformed Fourier system with $\eta = 2$ loses half an order, which is slightly improved for $\eta = 4$. In total we observe that some transformed Fourier systems are able to achieve the same decay rates as the Chebyshev system, when

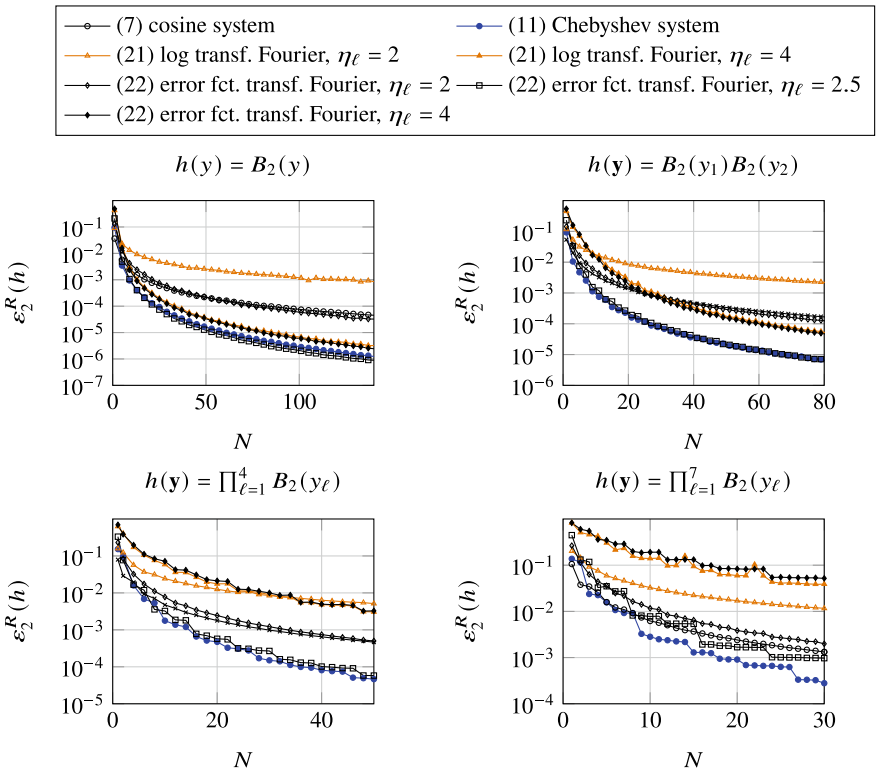


Fig. 4 Comparing the approximation errors $\varepsilon_2^R(h)$ of the tensored B_2-cutoff (27) approximated by various orthonormal systems in dimensions $d \in \{1, 2, 4, 7\}$

we use parameterized torus-to-cube transformations $\psi(\cdot, \eta)$ and pick an appropriate parameter $\eta \in \mathbb{R}_+$. The results are summarized in Table 2.

3.2 A Note on ℓ_∞-Approximation

As derived in [18] and recalled in (17), the transformed Fourier system (18) for non-periodic funtions is the result of applying an inverted change of variable $\psi^{-1}(\cdot, \boldsymbol{\eta})$ in the form of (15) to the Fourier system elements within the $L_2(\mathbb{T}^d)$-scalar product, in order to generate another orthonormal system in a given space $L_2\left([-\frac{1}{2}, \frac{1}{2}]^d, \omega\right)$. There are two interpretations for the resulting integral of the form

$$\int_{[0,1]^d} \frac{\varrho(\mathbf{y}, \boldsymbol{\eta})}{\omega(\mathbf{y})} \, \mathrm{e}^{2\pi \mathrm{i}(\mathbf{k}-\mathbf{m})\psi^{-1}(\mathbf{y},\boldsymbol{\eta})} \, \omega(\mathbf{y}) \, \mathrm{d}\mathbf{y} = \int_{\mathbb{T}^d} \mathrm{e}^{2\pi \mathrm{i}(\mathbf{k}-\mathbf{m})\mathbf{x}} \, \mathrm{d}\mathbf{x} = \delta_{\mathbf{k},\mathbf{m}}.$$

Table 2 The observed decay rates of the discrete approximation error $\varepsilon_2^R(h)$ as given in (28) when h is the univariate B_2-cutoff as defined in (26)

	Transformation	$\varepsilon_2^R(h)$
Equation (7)	Cosine system	$N^{-1.5}$
Equation (11)	Chebyshev system	$N^{-2.45}$
Equation (21)	Log transf. Fourier, $\eta = 2$	N^{-1}
Equation (21)	Log transf. Fourier, $\eta = 4$	$N^{-2.25}$
Equation (22)	Error fct. transf. Fourier, $\eta = 2$	$N^{-1.9}$
Equation (22)	Error fct. transf. Fourier, $\eta = 2.5$	$N^{-2.5}$
Equation (22)	Error fct. transf. Fourier, $\eta = 4$	$N^{-2.5}$

We either have another periodic system of the form $\left\{\mathrm{e}^{2\pi\mathrm{i}\mathbf{k}\cdot\psi^{-1}(\cdot,\boldsymbol{\eta})}\right\}_{\mathbf{k}\in I}$ and the weighted $L_2\left([0,1]^d, \varrho(\cdot,\boldsymbol{\eta})\right)$-scalar product; or we attach $\sqrt{\varrho(\cdot,\boldsymbol{\eta})/\omega(\cdot)}$ to the individual exponentials $\mathrm{e}^{2\pi\mathrm{i}\mathbf{k}\cdot\psi^{-1}(\cdot,\boldsymbol{\eta})}$ and end up with the non-periodic system (18) and the originally given weighted $L_2\left([-\frac{1}{2},\frac{1}{2}]^d, \omega\right)$-scalar product. If we consider a constant weight function $\omega \equiv 1$, then there is a drawback that comes with the later choice, because ϱ is unbounded and causes singularities at the boundary points of the elements in the approximated transformed Fourier sum (19). So, the pointwise approximation error ε_∞^R in (28) isn't finite, unless we consider a suitably weighted ℓ_∞-norm that counteracts the behavior of the approximant towards the boundary points, which is discussed more thoroughly in [18]. This strategy is based on choosing the weight function ω in such a way that the quotient $\varrho(\cdot,\boldsymbol{\eta})/\omega(\cdot)$ is either constant or converges at the boundary points. However, for any chosen torus-to-cube transformation—especially for the presented parameterized transformations $\psi(\cdot,\boldsymbol{\eta})$ in (21) and (22) with a fixed parameter $\boldsymbol{\eta}$—the weight function has to be chosen in such a way so that on one hand the singularities of the density function are controlled and on the other hand the given function h is still in $L_2([0,1]^d, \omega)$.

We achieve this effect for example by showing the connection of the transformed Fourier framework with the Chebyshev system, when we put the Chebyshev transformation (12) into the transformed Fourier system (18) despite the fact that it is not a torus-to-cube transformation as in (15). Considering the hyperbolic cross I_N^1 as defined in (23) and $x, y \in [0,1]$, we choose ψ to be the Chebyshev transformation (12) of the form $\psi(x) = \frac{1}{2} + \frac{1}{2}\cos\left(2\pi(x-\frac{1}{2})\right)$, with the inverse $\psi^{-1}(y) = \frac{1}{2} + \frac{\arccos(2y-1)}{2\pi}$ and the density $\varrho(y) = \frac{1}{2\pi\sqrt{y(1-y)}}$. By putting $\omega(y) = \varrho(y)$, the transformed Fourier system (18) turns into

$$\begin{aligned}\varphi_k(y) &= \mathrm{e}^{\pi\mathrm{i}k + \mathrm{i}k\arccos(2y)}\\ &= (-1)^k\left(\cos(k\arccos(2y-1)) + \mathrm{i}\sin(k\arccos(2y-1))\right)\end{aligned}$$

for $k \in \{-N, \ldots, N\}$ and by combining the positive and negative frequencies we obtain

$$\varphi_k(y) = \begin{cases} 1 & \text{for} \quad k = 0, \\ (-1)^k 2 \cos(k \arccos(2y-1)) & \text{for} \quad k \in \{1, 2, \ldots, N\}, \end{cases}$$

which is orthogonal with respect to the $L_2([0,1], \omega)$-scalar product with $\omega(y) = \frac{1}{2\pi\sqrt{y(1-y)}}$. With some additional scaling we obtain an orthonormal system that's equivalent to the Chebyshev system (11).

4 Conclusion

We considered the approximation of non-periodic functions on the cube $[0, 1]^d$ by different systems of orthonormal functions. We compared the Chebyshev system that is orthonormal with respect to a weighted L_2-scalar product, the system of half-periodic cosines that uses tent-transformed sampling nodes and a parameterized transformed Fourier system. For the cosine system, which basically only mirrors a non-periodic function at it's boundary points, as well as the transformed Fourier system with a small parameter, yielded the worst approximation errors. Switching to the Chebyshev system, which mirrors and additionally smoothens a given function, improved the approximation error decay. The same effect was obtained for the transformed Fourier system after increasing the parameter enough to obtain a better smoothening effect. The numerical experiments showcased the proposed parameter control in [18] that is set up by periodizing functions via families of parameterized torus-to-cube mappings. This approach in particular generalizes the idea used to derive Chebyshev polynomials.

Acknowledgements The authors thank the referees for their valuable suggestions and remarks. The first named author gratefully acknowledges the support by the funding of the European Union and the Free State of Saxony (ESF).

References

1. Adcock, B.: Modified Fourier Expansions: Theory, Construction and Applications (doctoral thesis) (2010). https://doi.org/10.17863/CAM.16096
2. Adcock, B.: Convergence acceleration of modified Fourier series in one or more dimensions. Math. Comput. **80**(273), 225–261 (2011)
3. Adcock, B., Iserles, A., Nørsett, S.P.: From high oscillation to rapid approximation II: expansions in Birkhoff series. IMA J. Numer. Anal. **32**(1), 105–140 (2012)
4. Bos, L., Caliari, M., De Marchi, S., Vianello, M., Xu, Y.: Bivariate Lagrange interpolation at the Padua points: the generating curve approach. J. Approx. Theory **143**(1), 15–25 (2006). https://doi.org/10.1016/j.jat.2006.03.008, https://www.sciencedirect.com/science/article/pii/S0021904506000505. Special Issue on Foundations of Computational Mathematics

5. Boyd, J.P.: Chebyshev and Fourier Spectral Methods, 2nd edn. Dover Press, New York, NY, USA (2000)
6. Byrenheid, G., Kämmerer, L., Ullrich, T., Volkmer, T.: Tight error bounds for rank-1 lattice sampling in spaces of hybrid mixed smoothness. Numer. Math. **136**, 993–1034 (2017). https://doi.org/10.1007/s00211-016-0861-7
7. Cools, R., Kuo, F.Y., Nuyens, D.: Constructing lattice rules based on weighted degree of exactness and worst case error. Computing **87**, 63–89 (2010)
8. Cools, R., Kuo, F.Y., Nuyens, D., Suryanarayana, G.: Tent-transformed lattice rules for integration and approximation of multivariate non-periodic functions. J. Complex. **36**, 166–181 (2016)
9. Dencker, P., Erb, W.: Multivariate polynomial interpolation on Lissajous-Chebyshev nodes. J. Approx. Theory **219**, 15–45 (2017)
10. Dick, J., Nuyens, D., Pillichshammer, F.: Lattice rules for nonperiodic smooth integrands. Numer. Math. **126**, 259–291 (2014). https://doi.org/10.1007/s00211-013-0566-0
11. Goda, T., Suzuki, K., Yoshiki, T.: Lattice rules in non-periodic subspaces of Sobolev spaces. Numer. Math. **141**(2), 399–427 (2019). https://doi.org/10.1007/s00211-018-1003-1
12. Irrgeher, C., Kritzer, P., Pillichshammer, F.: Integration and approximation in cosine spaces of smooth functions. Math. Comput. Simul. **143**, 35–45 (2018)
13. Iserles, A., Nørsett, S.: From high oscillation to rapid approximation I: Modified Fourier expansions. IMA J. Numer. Anal **28**, 862–887 (2008). https://doi.org/10.1093/imanum/drn006
14. Kämmerer, L.: High Dimensional Fast Fourier Transform Based on Rank-1 Lattice Sampling. Dissertation. Universitätsverlag Chemnitz (2014). http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-157673
15. Kämmerer, L., Potts, D., Volkmer, T.: Approximation of multivariate periodic functions by trigonometric polynomials based on rank-1 lattice sampling. J. Complex. **31**, 543–576 (2015)
16. Kuo, F., Migliorati, G., Nobile, F., Nuyens, D.: Function integration, reconstruction and approximation using rank-1 lattices. Math. Comput. **90**(330), 1861–1897 (2021)
17. Nasdala, R., Potts, D.: Transformed rank-1 lattices for high-dimensional approximation. Electron. Trans. Numer. Anal. **53**, 239–282 (2020). https://doi.org/10.1553/etna_vol53s239
18. Nasdala, R., Potts, D.: Efficient multivariate approximation on the cube. Numer. Math. **147**(2), 393–429 (2021)
19. Potts, D., Volkmer, T.: Fast and exact reconstruction of arbitrary multivariate algebraic polynomials in Chebyshev form. In: 11th International Conference on Sampling Theory and Applications (SampTA 2015), pp. 392–396 (2015)
20. Shen, J., Tang, T., Wang, L.L.: Spectral methods. In: Springer Series in Computational Mathematics, vol. 41. Springer, Berlin (2011)
21. Sloan, I.H., Kachoyan, P.J.: Lattice methods for multiple integration: Theory, error analysis and examples. SIAM J. Numer. Anal. **24**, 116–128 (1987)
22. Suryanarayana, G., Nuyens, D., Cools, R.: Reconstruction and collocation of a class of non-periodic functions by sampling along tent-transformed rank-1 lattices. J. Fourier Anal. Appl. **22**, 187–214 (2016)
23. Temlyakov, V.N.: Reconstruction of periodic functions of several variables from the values at the nodes of number-theoretic nets. Anal. Math. **12**, 287–305 (1986). https://doi.org/10.1007/BF01909367. In Russian
24. Volkmer, T.: Multivariate Approximation and High-Dimensional Sparse FFT Based on Rank-1 Lattice Sampling. Dissertation. Universitätsverlag Chemnitz (2017). http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-222820

Applications of Multivariate Quasi-Random Sampling with Neural Networks

Marius Hofert, Avinash Prasad, and Mu Zhu

Abstract Generative moment matching networks (GMMNs) are suggested for modeling the cross-sectional dependence between stochastic processes. The stochastic processes considered are geometric Brownian motions and ARMA–GARCH models. Geometric Brownian motions lead to an application of pricing American basket call options under dependence and ARMA–GARCH models lead to an application of simulating predictive distributions. In both types of applications the benefit of using GMMNs in comparison to parametric dependence models is highlighted and the fact that GMMNs can produce dependent quasi-random samples with no additional effort is exploited to obtain variance reduction.

Keywords Generative moment matching networks · Copulas · Quasi-random sampling · American basket option pricing · ARMA–GARCH · Predictive distributions

1 Introduction

Given data $\boldsymbol{X}_1, \dots, \boldsymbol{X}_{n_{\text{trn}}} \sim F_{\boldsymbol{X}}$ in $\mathbb{R}^d$, a fundamental statistical task is to learn something about the distribution $F_{\boldsymbol{X}}$ itself. Traditionally, this is done by assuming $F_{\boldsymbol{X}}$ to belong to a certain parametric family, say, $F_{\boldsymbol{X}}(\cdot;\boldsymbol{\theta})$, and estimating the parameter vector $\boldsymbol{\theta}$ from data, e.g., by maximum likelihood.

Alternatively, we can learn to generate samples from $F_{\boldsymbol{X}}$ directly. Specifically, given a sample $\boldsymbol{V}_1, \dots, \boldsymbol{V}_{n_{\text{gen}}} \sim F_{\boldsymbol{V}}$ in $\mathbb{R}^d$, where $F_{\boldsymbol{V}}$ is a simple distribution (e.g., the independent standard normal or standard uniform), can we learn a generator

M. Hofert (✉) · A. Prasad · M. Zhu
Department of Statistics and Actuarial Science, University of Waterloo, 200 University Avenue West, N2L 3G1 Waterloo, ON, USA
e-mail: marius.hofert@uwaterloo.ca

A. Prasad
e-mail: a2prasad@uwaterloo.ca

M. Zhu
e-mail: mu.zhu@uwaterloo.ca

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_14

$G(\cdot)$ such that $G(\boldsymbol{V}_1), \dots, G(\boldsymbol{V}_{n_{\text{gen}}}) \sim F_{\boldsymbol{X}}$? An answer to this question has been provided recently by the machine learning community [2, 9]: yes, we can do so using a so-called generative moment matching network (GMMN).

We have found [6] that GMMNs, once trained, also allow us to generate quasi-random samples (QRS) from $F_{\boldsymbol{X}}$ "for free". This is exciting because we thus have a universal and flexible approach for constructing cross-sectionally dependent QRS from a large variety of different models, including parametric models but especially also empirical ones (for which the true underlying dependence model is unknown). In the present paper, we will first give a quick overview of what GMMNs are, and how they can be used to generate QRS from almost any distribution. We will then focus on two applications: pricing American basket call options, and making probabilistic forecasts for multivariate time series.

Throughout the paper, we rely heavily on the decomposition afforded by Sklar's Theorem [16], namely,

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = C(F_{X_1}(x_1), \dots, F_{X_d}(x_d)), \quad \boldsymbol{x} = (x_1, \dots, x_d) \in \mathbb{R}^d,$$

where $C : [0,1]^d \mapsto [0,1]$ is the unique underlying copula [3, 12], and $F_{X_1}, \dots, F_{x_d}$ are the continuous marginal distributions of $X_1, \dots, X_d$. This allows us to focus on the problem of generating $\boldsymbol{U}_k = G(\boldsymbol{V}_k) \sim C$, $k = 1, \dots, n_{\text{gen}}$, from which we can simply obtain $\boldsymbol{X}_k = F_{\boldsymbol{X}}^{-1}(\boldsymbol{U}_k) \sim F_{\boldsymbol{X}}$, $k = 1, \dots, n_{\text{gen}}$, where $F_{\boldsymbol{X}}^{-1}(\boldsymbol{u}) \equiv (F_{X_1}^{-1}(u_1), \dots, F_{X_d}^{-1}(u_d))$.

For both applications we present in this paper, we have not just static data $\boldsymbol{X} \sim F_{\boldsymbol{X}}$ but realizations of a stretch of a stochastic process $\boldsymbol{X}_t$ over time, with

$$\boldsymbol{X}_{t_k} = \eta(\boldsymbol{Z}_k \mid \boldsymbol{X}_{t_1}, \dots, \boldsymbol{X}_{t_{k-1}}), \quad k = 1, \dots, n_{\text{trn}}, \tag{1}$$

where $\boldsymbol{Z}_1, \dots, \boldsymbol{Z}_{n_{\text{trn}}} \overset{\text{ind.}}{\sim} F_{\boldsymbol{Z}}$ and η is "decomposable" into component-wise or marginal functions η_j, in the sense that, for each $j = 1, \dots, d$, we have

$$X_{t_k,j} = \eta_j(Z_{k,j} \mid X_{t_1,j}, \dots, X_{t_{k-1},j}). \tag{2}$$

The key structure here is that, for any fixed j, $Z_{1,j}, \dots, Z_{n_{\text{trn}},j} \overset{\text{ind.}}{\sim} F_{Z_j}$ but, for any fixed k, $Z_{k,1}, \dots, Z_{k,d}$ are dependent. For each j, the function η_j allows us to describe the marginal stochastic processes $X_{t,j}$ as a transformation of iid random variables $Z_{t,j}$. In the two applications we consider, η_j can be viewed as a function of the conditional mean and variance processes; see Sects. 3.1 and 4.1 for details. However, the functional form of η_j can be fairly general and the only restriction is that it is invertible.

Conceptually, we may think of a Sklar decomposition at every time point t,

$$F_{\boldsymbol{X}_{t_k}}(\boldsymbol{x}) = C(F_{X_{t_k,1}}(x_1), \dots, F_{X_{t_k,d}}(x_d)),$$

where the copula C remains constant over time and hence the same at all t, but the marginal distributions $F_{X_{t_k,1}}, \ldots, F_{X_{t_k,d}}$ may vary over time. And the operation

$$F^{-1}_{\boldsymbol{X}_{t_k}}(\boldsymbol{U}_k) = (F^{-1}_{X_{t_k,1}}(U_{k,1} \,|\, \mathcal{F}_{t_{k-1},1}), \ldots, F^{-1}_{X_{t_k,d}}(U_{k,d} \,|\, \mathcal{F}_{t_{k-1},d}))$$

is now conditional on the entire history of the process up to and including time t_{k-1}, denoted here by the natural filtration $\mathcal{F}_{t_{k-1}}$, where, for any given s, $\mathcal{F}_{s,j} = \sigma(\{X_{s',j} : s' \le s\})$ for all $j = 1, \ldots, d$. Under (1)–(2), the component-wise conditional operation $F^{-1}_{X_{t_k,j}}(\cdot \,|\, \mathcal{F}_{t_{k-1},j})$ is simply

$$F^{-1}_{X_{t_k,j}}(U_{k,j} \,|\, \mathcal{F}_{t_{k-1},j}) = \eta_j(F^{-1}_{Z_j}(U_{k,j}) \,|\, X_{t_1,j}, \ldots, X_{t_{k-1},j}).$$

For American basket call options, each $F_{X_{t_k,j}}(\cdot \,|\, \mathcal{F}_{t_{k-1},j})$ is dictated by an underlying geometric Brownian motion (GBM). For multivariate time series, each $F_{X_{t_k,j}}(\cdot \,|\, \mathcal{F}_{t_{k-1},j})$ is dictated by an underlying ARMA–GARCH process [1, 17].

To price American basket call options as well as to make probabilistic forecasts for multivariate time series, the key lies in repeatedly simulating the time path forward for each $X_{t,j}$. For any given path i and time point t_k, this is done by generating

$$\hat{X}^{(i)}_{t_k,j} = \hat{F}^{-1}_{X_{t_k,j}}\big(U^{(i)}_{k,j} \,|\, \mathcal{F}^{(i)}_{t_{k-1},j}\big) = \hat{\eta}_j\big(\hat{F}^{-1}_{Z_j}(U^{(i)}_{k,j}) \,|\, \hat{X}^{(i)}_{t_1,j}, \ldots, \hat{X}^{(i)}_{t_{k-1},j}\big), \tag{3}$$

where $\hat{X}^{(i)}_{s,j} = X_{s,j}$ across all i if time point s is part of the training set. Typically, each $\boldsymbol{U}^{(i)}_k = (U^{(i)}_{k,1}, \ldots, U^{(i)}_{k,d})^\top$ is generated from a parametric copula model, whereas we propose to generate it nonparametrically from a GMMN fitted to the training data.

2 GMMNs, Pseudo-Random and Quasi-Random Sampling

As established in Sect. 1, we focus on generating $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_{n_{\text{gen}}}$ from the underlying copula C. To do so, we rely on training data $\hat{\boldsymbol{U}}_1, \ldots, \hat{\boldsymbol{U}}_{n_{\text{trn}}}$. These can be obtained by first estimating and then removing the marginal distributions from $\boldsymbol{X}_{t_k}$. For our two applications, this is achieved by

$$\hat{U}_{k,j} = \hat{F}_{X_{t_k,j}}(X_{t_k,j} \,|\, \mathcal{F}_{t_{k-1},j}) = \hat{F}_{Z_j}\big(\hat{\eta}^{-1}_j(X_{t_k,j} \,|\, X_{t_1,j}, \ldots, X_{t_{k-1},j})\big) \tag{4}$$

under (1)–(2).

2.1 Generative Moment Matching Networks (GMMNs)

Let $\mathcal{G}$ denote a set of neural networks with a pre-determined architecture. A GMMN is the solution to the following minimization problem:

$$\min_{G\in\mathcal{G}} \frac{1}{n_{\text{trn}}^2}\sum_{k=1}^{n_{\text{trn}}}\sum_{k'=1}^{n_{\text{trn}}} K(\hat{\boldsymbol{U}}_k, \hat{\boldsymbol{U}}_{k'}) - \frac{2}{n_{\text{trn}}n_{\text{gen}}}\sum_{k=1}^{n_{\text{trn}}}\sum_{k'=1}^{n_{\text{gen}}} K(\hat{\boldsymbol{U}}_k, G(\boldsymbol{V}_{k'})) + \frac{1}{n_{\text{gen}}^2}\sum_{k=1}^{n_{\text{gen}}}\sum_{k'=1}^{n_{\text{gen}}} K(G(\boldsymbol{V}_k), G(\boldsymbol{V}_{k'})), \quad (5)$$

where $K(\boldsymbol{u}, \boldsymbol{v})$ is a kernel function, such as the Gaussian or radial basis kernel. The minimizer of (5) ensures the distribution of the generated sample $\{G(\boldsymbol{V}_{k'})\}_{k'=1}^{n_{\text{gen}}}$ is as close as possible to that of the training sample $\{\hat{\boldsymbol{U}}_k\}_{k=1}^{n_{\text{trn}}}$. This is because the criterion being minimized in (5) is equal to

$$\left\| \frac{1}{n_{\text{trn}}}\sum_{k=1}^{n_{\text{trn}}} \varphi(\hat{\boldsymbol{U}}_k) - \frac{1}{n_{\text{gen}}}\sum_{k'=1}^{n_{\text{gen}}} \varphi(G(\boldsymbol{V}_{k'})) \right\|^2,$$

where $\varphi(\cdot)$ is the implied feature map of K such that $K(\boldsymbol{u}, \boldsymbol{v}) = \varphi(\boldsymbol{u})^\top\varphi(\boldsymbol{v})$ and, for the Gaussian kernel and its implied feature map, the two statistics — $(1/n_{\text{trn}})\sum_{k=1}^{n_{\text{trn}}}\varphi(\hat{\boldsymbol{U}}_k)$ and $(1/n_{\text{gen}})\sum_{k'=1}^{n_{\text{gen}}}\varphi(G(\boldsymbol{V}_{k'}))$ — contain all empirical moments of the training sample $\{\hat{\boldsymbol{U}}_k\}_{k=1}^{n_{\text{trn}}}$ and the generated sample $\{G(\boldsymbol{V}_{k'})\}_{k'=1}^{n_{\text{gen}}}$, respectively. This is also where the name GMMN comes from.

Here, we will not go into any more details of how the optimization problem (5) is actually solved; instead, we simply refer the reader to [2, 6, 9]. A very short summary is "by stochastic gradient descent" but there are many practical details such as the need for mini-batch optimization, and the use of a mixture (rather than a single) kernel function, and so on.

2.2 Pseudo-Random Sampling

After having trained a GMMN, generating n_{pth} paths of n_{gen} d-dimensional pseudo-random samples from it can be done as follows.

2.3 Quasi-Random Sampling

As mentioned in Sect. 1, we have found in [6] that GMMNs can preserve low discrepancy—and thus achieve a variance reduction effect—if fed with quasi-

Algorithm 1: Pseudo-random sampling of GMMN-dependent paths

1. Fix n_{pth}, the number of paths, and n_{gen}, the number of d-dimensional samples to be generated for each path.
2. For $i = 1, \dots, n_{\text{pth}}$, $k = 1, \dots, n_{\text{gen}}$, draw $\boldsymbol{V}_k^{(i)} = (V_{k,1}^{(i)}, \dots, V_{k,d}^{(i)}) \overset{\text{ind.}}{\sim} F_{\boldsymbol{V}}$, for example, via $\boldsymbol{V}_k^{(i)} = F_{\boldsymbol{V}}^{-1}(\boldsymbol{U}_k'^{(i)})$, where $\boldsymbol{U}_k'^{(i)} = (U_{k,1}'^{(i)}, \dots, U_{k,d}'^{(i)}) \overset{\text{ind.}}{\sim} \mathrm{U}(0,1)^d$.
3. Return the pseudo-observations of $\boldsymbol{U}_k^{(i)} = G(\boldsymbol{V}_k^{(i)})$, $i = 1, \dots, n_{\text{pth}}$, $k = 1, \dots, n_{\text{gen}}$.

random samples. In Algorithm 1 above, we can simply replace $\boldsymbol{U}_k'^{(i)} \overset{\text{ind.}}{\sim} \mathrm{U}(0,1)^d$, for $i = 1, \dots, n_{\text{pth}}$ and $k = 1, \dots, n_{\text{gen}}$, by a randomized quasi-Monte Carlo (RQMC) point set, such as a randomized Sobol' sequence; see [6] for empirical evidence under a great variety of multivariate distributions (specifically, copulas). As in [8, Sect. 7.3], we generate the RQMC point set in a specific way. To this end, let $d^* = n_{\text{gen}} \cdot d$. We then generate an RQMC point set $\tilde{P}_{n_{\text{pth}}} = \{\tilde{\boldsymbol{v}}_1, \dots, \tilde{\boldsymbol{v}}_{n_{\text{pth}}}\}$ of n_{pth} d^*-dimensional points, resulting in an (n_{pth}, d^*)-matrix, whose columns are blocked in n_{gen} groups of size d each to form n_{pth} paths of n_{gen} d-dimensional quasi-random samples from the trained GMMN. We thus obtain the following algorithm.

Algorithm 2: Quasi-random sampling of GMMN-dependent paths

1. Fix n_{pth}, the number of paths, and n_{gen}, the number of d-dimensional samples to be generated for each path. Furthermore, set $d^* = n_{\text{gen}} \cdot d$.
2. Compute a d^*-dimensional RQMC point set $\tilde{P}_{n_{\text{pth}}} = \{\tilde{\boldsymbol{v}}_1, \dots, \tilde{\boldsymbol{v}}_{n_{\text{pth}}}\}$, for example, as a randomized Sobol' sequence.
3. Compute $\boldsymbol{V}_k^{(i)} = F_{\boldsymbol{V}}^{-1}(\tilde{v}_{i,(k-1)d+1}, \dots, \tilde{v}_{i,kd})$, $i = 1, \dots, n_{\text{pth}}$, $k = 1, \dots, n_{\text{gen}}$.
4. Return the pseudo-observations of $\boldsymbol{U}_k^{(i)} = G(\boldsymbol{V}_k^{(i)})$, $i = 1, \dots, n_{\text{pth}}$, $k = 1, \dots, n_{\text{gen}}$.

3 American Basket Option Pricing

In this section, we demonstrate the usefulness of GMMNs and the dependent quasi-random samples they can generate for pricing American basket call options.

3.1 Model

The basket portfolio consists of d assets whose prices $X_{t,1}, \dots, X_{t,d}$ follow geometric Brownian motions. With risk-neutral drift, $(X_{t,j})_{t \geq 0}$ can be represented as

$$X_{t,j} = X_{0,j} \exp(Y_{t,j}) \quad \text{for} \quad Y_{t,j} = (r - \sigma_j^2/2)t + \sigma_j W_{t,j}, \tag{6}$$

where r is the risk-free interest rate, σ_j is the volatility parameter of the jth asset (quantified as the standard deviation of $Y_{t,j}$ over one unit of time) and $W_{t,j}$ is a standard Wiener process. A discretization of $W_{t,j}$ on $0 = t_0 < t_1 < \cdots < t_{n_{\text{trn}}}$ is given by

$$W_{t_k,j} = \sum_{l=1}^{k} \sqrt{t_l - t_{l-1}} Z_{l,j}, \quad Z_{1,j}, \dots, Z_{n_{\text{trn}},j} \overset{\text{ind.}}{\sim} \mathrm{N}(0,1), \tag{7}$$

and thus a discretization of (6) is given by

$$X_{t_k,j} = X_{t_0,j} \exp(Y_{t_k,j}) \quad \text{for} \quad Y_{t_k,j} = (r - \sigma_j^2/2)t_k + \sigma_j \sum_{l=1}^{k} \sqrt{t_l - t_{l-1}} Z_{l,j}; \tag{8}$$

this is in line with Equation (2). Dependence between $X_{t_k,1}, \dots, X_{t_k,d}$ is introduced by making the increments $Z_{k,1}, \dots, Z_{k,d}$ dependent either through a parametric copula C_{PM} (in which case we speak of a *copula–GBM model* for the joint stock price process $\boldsymbol{X}_{t_k} = (X_{t_k,1}, \dots, X_{t_k,d})$, $k = 0, \dots, n_{\text{trn}}$) or through a GMMN (in which case we speak of a *GMMN–GBM model*).

3.2 Estimation

Suppose that for each $j = 1, \dots, d$, we have $n_{\text{trn}} + 1$ realizations $X_{t_k,j}$, $k = 0, \dots, n_{\text{trn}}$. We fix the risk-free interest rate to be r and estimate σ_j as the sample standard deviation $\hat{\sigma}_j$ of the log-returns $Y_{t_k,j} = \log(X_{t_k,j}/X_{t_{k-1},j})$, $k = 1, \dots, n_{\text{trn}}$. We then recover the realizations $\hat{Z}_{1,j}, \dots, \hat{Z}_{n_{\text{trn}},j}$ from $X_{t_0,j}, \dots, X_{t_{n_{\text{trn}}},j}$ via (7) and (8) by

$$\begin{aligned}
\hat{W}_{t_k,j} &= \frac{1}{\hat{\sigma}_j} \left(\log\left(\frac{X_{t_k,j}}{X_{t_0,j}} \right) - \left(r - \frac{\hat{\sigma}_j^2}{2} \right) t_k \right), \quad k = 0, \dots, n_{\text{trn}}, \\
\hat{Z}_{k,j} &= \frac{\hat{W}_{t_k,j} - \hat{W}_{t_{k-1},j}}{\sqrt{t_k - t_{k-1}}}, \quad k = 1, \dots, n_{\text{trn}},
\end{aligned}$$

a process we refer to as *deBrowning* (analogously to deGARCHing known for ARMA–GARCH processes); this is in line with Eq. (4). The cross-sectional depen-

dence of the iid $\hat{\boldsymbol{Z}}_k = (\hat{Z}_{k,1}, \ldots, \hat{Z}_{k,d})$, $k = 1, \ldots, n_{\text{trn}}$, with supposedly standard normal margins is then modeled based on the *pseudo-observations*

$$\hat{U}_{k,j} = \frac{\hat{R}_{k,j}}{n_{\text{trn}} + 1}, \quad k = 1, \ldots, n_{\text{trn}}, \ j = 1, \ldots, d, \tag{9}$$

where $\hat{R}_{k,j}$ denotes the rank of $\hat{Z}_{k,j}$ among $\hat{Z}_{1,j} \ldots, \hat{Z}_{n_{\text{trn}},j}$. Note that using the pseudo-observations instead of assuming standard normality of the margins of $\hat{Z}_k = (\hat{Z}_{k,1}, \ldots, \hat{Z}_{k,d})$ reduces the effect of a potential misspecification of the margins on the estimation of the cross-sectional dependence; see [4]. As cross-sectional dependence model for the distribution of $\hat{\boldsymbol{U}}_k = (\hat{U}_{k,1}, \ldots, \hat{U}_{k,d})$, $k = 1, \ldots, n_{\text{trn}}$, we use either a fitted parametric copula $\hat{C}_{\text{PM}}$ or a trained GMMN $G : \mathbb{R}^d \to [0, 1]^d$. Analogously to $\hat{C}_{\text{PM}}$, we denote the copula of the samples generated from the trained GMMN by $\hat{C}_{\text{NN}}$ and view them as an approximation to the target dependence structure of $\hat{\boldsymbol{U}}_1, \ldots, \hat{\boldsymbol{U}}_{n_{\text{trn}}}$. In our option pricing application, we compare the trained GMMN with fitted Clayton, normal and t copulas — with unstructured correlation matrices for the latter two. We also include the independence copula as benchmark.

3.3 Simulation

For simulating the dependent asset prices, samples from $\hat{C}_{\text{NN}}$ and $\hat{C}_{\text{PM}}$ are mapped to N(0, 1) margins to obtain samples from the joint increment distribution and thus, after undoing deBrowning, to obtain samples from the dependent asset prices. Algorithm 3 describes these steps for simulating n_{pth}-many paths of $\boldsymbol{X}_{t_k} = (X_{t_k,1}, \ldots, X_{t_k,d})$, $k = 0, \ldots, n_{\text{gen}}$, for our newly proposed GMMN–GBM model. The dependent asset price processes then serve as inputs for computing the present value of the American basket call option considered.

3.4 Application

To price an American basket call option, we assume the option can be exercised at $t_1, \ldots, t_{n_{\text{gen}}}$ and that $t_{n_{\text{gen}}} = T$ is the maturity (or expiry) of the option contract. We are then interested in estimating

$$\max_{k=1,\ldots,n_{\text{gen}}} \mathbb{E}(\exp(-rt_k)H(\boldsymbol{X}_{t_k}, t_k)), \tag{10}$$

where $H(\boldsymbol{X}_{t_k}, t_k) = \max\left\{\left((1/d)\sum_{j=1}^{d} X_{t_k,j}\right) - K, 0\right\}$ is the payoff function for the American basket call option with strike price K; note that the expectation in (10) is with respect to the risk-neutral measure. To find a solution of (10), we follow

Algorithm 3: Pseudo- and quasi-random sampling of GMMN-GBM paths

1. Fix n_{pth}, the number of paths, and n_{gen}, the number (or total time steps) of d-dimensional samples to be generated for each path. Furthermore, fix the risk-free interest rate r, the estimated asset price volatilities $\hat{\sigma}_j$, $j = 1, \dots, d$, and the initial asset prices $\hat{\boldsymbol{X}}_{t_0}^{(i)} = \boldsymbol{X}_{t_{n_{\text{trn}}}}$, $i = 1, \dots, n_{\text{pth}}$.
2. Generate $\boldsymbol{U}_k^{(i)} = (U_{k,1}^{(i)}, \dots, U_{k,d}^{(i)})$, $i = 1, \dots, n_{\text{pth}}$, $k = 1, \dots, n_{\text{gen}}$, according to Algorithm 1 (for pseudo-random samples) or Algorithm 2 (for quasi-random samples).
3. Compute the increments $Z_k^{(i)} = (Z_{k,1}^{(i)}, \dots, Z_{k,d}^{(i)})$ with $Z_{k,j}^{(i)} = \Phi^{-1}(U_{k,j}^{(i)})$ for $i = 1, \dots, n_{\text{pth}}$, $k = 1, \dots, n_{\text{gen}}$, where Φ^{-1} is the N(0, 1) quantile function.
4. Compute $\hat{\boldsymbol{X}}_{t_k}^{(i)} = (\hat{X}_{t_k,1}^{(i)}, \dots, \hat{X}_{t_k,d}^{(i)})$ with

$$\hat{X}_{t_k,j}^{(i)} = \hat{X}_{t_0,j}^{(i)} \exp\left((r - \hat{\sigma}_j^2/2) t_k + \hat{\sigma}_j \sum_{l=1}^{k} \sqrt{t_l - t_{l-1}} Z_{l,j}^{(i)} \right)$$

   for $i = 1, \dots, n_{\text{pth}}$, $k = 1, \dots, n_{\text{gen}}$; see (8).
5. Return $\hat{\boldsymbol{X}}_{t_k}^{(i)}$, $i = 1, \dots, n_{\text{pth}}$, $k = 1, \dots, n_{\text{gen}}$.

the dynamic programming principle and traverse each simulated path of the underlying asset price process backwards in time (starting at maturity) while making hold/exercise decisions at each time point. To make such decisions along the ith path, we need to compare the *exercise value* $H(\boldsymbol{X}_{t_k}^{(i)}, t_k)$ with the expected *continuation value*. To this end, we work with the least squares Monte Carlo algorithm of [10] according to which the expected continuation value is estimated by regressing the realized option payoffs from continuation on basis functions of the basket price. As basis functions we use the first three weighted Laguerre polynomials

$$L_0(x) = e^{-x/2}, \quad L_1(x) = e^{-x/2}(1 - x^2/2), \quad L_2(x) = e^{-x/2}(1 - 2x + x^2/2),$$

where x is the current asset price, and we also use their corresponding three cross-products $L_0(x)L_1(x)$, $L_0(x)L_2(x)$ and $L_1(x)L_2(x)$. Algorithm 4 provides the details; see also [10].

As data application we consider daily adjusted closing prices of 10 S&P 500 constituents from 1995-01-01 to 2015-12-31. The selected constituents include three stocks from the information technology sector (Intel Corp. (INTC), Oracle Corp. (ORCL) and International Business Machines Corp. (IBM)), three stocks from the financial sector (Capital One Financial Corp. (COF), JPMorgan Chase & Co. (JPM) and American International Group Inc (AIG)) and four stocks from the industrial sector (3M Company (MMM), Boeing Company (BA), General Electric (GE) and Caterpillar Inc. (CAT)). In addition, we also consider sub-portfolios of these constituents with dimensions $d = 5$ (consisting of INTC, ORCL, IBM, COF and AIG) and $d = 3$ (consisting of INTC, IBM and AIG). The data used for this application

Algorithm 4: Least squares Monte Carlo for American basket call option pricing

1. Simulate asset price paths $\hat{\boldsymbol{X}}_{t_k}^{(i)}$, $i = 1, \ldots, n_{\text{pth}}$, $k = 1, \ldots, n_{\text{gen}}$, using Algorithm 3 based on either GMMN pseudo-random samples or GMMN quasi-random samples.
2. Compute the value of the American basket call option at maturity: $\hat{V}_{t_{n_{\text{gen}}}}^{(i)} = H(\hat{\boldsymbol{X}}_{t_{n_{\text{gen}}}}^{(i)}, t_{n_{\text{gen}}})$, $i = 1, \ldots, n_{\text{pth}}$.
3. For $k = n_{\text{gen}} - 1, \ldots, 1$ do:
   a. Compute the discounted values $\hat{V}_{t_k}^{(i)} = \exp(-r(t_k - t_{k-1}))\hat{V}_{t_{k+1}}^{(i)}$, $i = 1, \ldots, n_{\text{pth}}$.
   b. Compute the basket price $\bar{\hat{X}}_{t_k}^{(i)} = \frac{1}{d}\sum_{j=1}^{d} \hat{X}_{t_k,j}^{(i)}$, $i = 1, \ldots, n_{\text{pth}}$.
   c. Compute the design matrix $D_{t_k} \in \mathbb{R}^{n_{\text{pth}} \times 7}$ with ith row given by $(1, L_0(\bar{\hat{X}}_{t_k}^{(i)}), L_1(\bar{\hat{X}}_{t_k}^{(i)}), L_2(\bar{\hat{X}}_{t_k}^{(i)}), L_0(\bar{\hat{X}}_{t_k}^{(i)})L_1(\bar{\hat{X}}_{t_k}^{(i)}), L_0(\bar{\hat{X}}_{t_k}^{(i)})L_2(\bar{\hat{X}}_{t_k}^{(i)}), L_1(\bar{\hat{X}}_{t_k}^{(i)})L_2(\bar{\hat{X}}_{t_k}^{(i)}))$, $i = 1, \ldots, n_{\text{pth}}$.
   d. Regress $\hat{V}_{t_k}^{(i)}$, $i = 1, \ldots, n_{\text{pth}}$, on D_{t_k} and use the fitted values of the regression as continuation values, i.e., $\hat{C}_{t_k}^{(i)} = D_{t_k}(D_{t_k}' D_{t_k})^{-1} D_{t_k}' \hat{V}_{t_k}^{(i)}$, $i = 1, \ldots, n_{\text{pth}}$. Adjust the continuation values for bias by setting $\hat{C}_{t_k}^{(i)} = \max\{\hat{C}_{t_k}^{(i)}, 0\}$, $i = 1, \ldots, n_{\text{pth}}$; see [10].
   e. Compute the exercise values $\hat{E}_{t_k}^{(i)} = H\big(\hat{\boldsymbol{X}}_{t_k}^{(i)}, t_k\big)$, $i = 1, \ldots, n_{\text{pth}}$.
   f. Set $\hat{V}_{t_k}^{(i)} = \hat{E}_{t_k}^{(i)}$ for all $i = 1, \ldots, n_{\text{pth}}$ such that $\hat{E}_{t_k}^{(i)} \geq \hat{C}_{t_k}^{(i)}$.
4. Compute the American basket call option price as $e^{-r(t_1 - t_0)} \frac{1}{n_{\text{pth}}} \sum_{i=1}^{n_{\text{pth}}} \hat{V}_{t_1}^{(i)}$.

can be obtained from the R package `qrmdata`. We choose the risk-free interest rate to be $r = 0.0005$ (annualized), in line with the US treasury bond yield rates at the onset of the option contract for maturities similar to the lengths of the contracts considered; the maturities in our pricing exercise range from one day to 100 days. When selecting the strike price K for the various basket options with different maturities, we try to ensure that the values of these basket call options are not too close to zero by selecting K to be close to the current market value of the basket—specifically, 101% of the basket value at time point $t_{n_{\text{trn}}}$.

To model the pseudo-observations (9) of each of the the three portfolios considered, we use parametric copulas C_{PM} that are known to capture the dependence between financial return series well, such as the normal and the t. We also consider a Clayton copula which is lower tail dependent. The choice of the normal and the t is also because we are able to generate *quasi*-random samples from them, while quasi-random sampling is not readily available for all parametric copula models; see [6]. This allows us to compare the variance reduction factors achieved when pricing American basket call options with the GMMN versus parametric copulas such as the normal and the t. For GMMNs, we use the same architecture and choice of hyperparameters as described in [6]. All parametric copulas are fitted using the maximum pseudo-likelihood method; see [5, Sect. 4.1.2]. For a detailed comparison of run time measurements between GMMNs and parametric copulas in the contexts of training

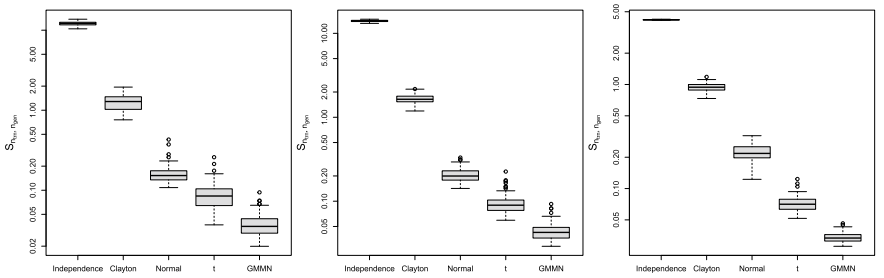


Fig. 1 Box plots based on $n_{\text{rep}} = 100$ realizations of $S_{n_{\text{trn}},n_{\text{gen}}}$ computed for portfolios of dimensions $d = 3$ (left), $d = 5$ (middle) and $d = 10$ (right) with training sample size $n_{\text{trn}} = 5287$. For each fitted dependence model we generate a pseudo-random sample of size $n_{\text{gen}} = 10\,000$. From these box plots, we can see that the GMMNs provide a much better fit than well-known parametric copulas for all three portfolios considered

(or model fitting), as well as for pseudo-random and quasi-random sampling, refer to [6, Appendix B].

To evaluate the fit of a dependence model, we use a Cramér-von-Mises type test statistic introduced by [14] to assess the equality of two empirical copulas; see [6]. This statistic is defined as

$$S_{n_{\text{trn}},n_{\text{gen}}} = \frac{1}{\sqrt{\frac{1}{n_{\text{trn}}} + \frac{1}{n_{\text{gen}}}}} \int_{[0,1]^d} \left(C_{n_{\text{trn}}}(\boldsymbol{u}) - C_{n_{\text{gen}}}(\boldsymbol{u})\right)^2 \mathrm{d}\boldsymbol{u}, \tag{11}$$

where $C_{n_{\text{trn}}}(\boldsymbol{u})$ is the empirical copula of the n_{trn} pseudo-observations used to fit the dependence model (see (9)) and $C_{n_{\text{gen}}}(\boldsymbol{u})$ is the empirical copula of the n_{gen} samples generated from the fitted dependence model (either $\hat{C}_{\text{PM}}$ or $\hat{C}_{\text{NN}}$). Figure 1 shows box plots of $S_{n_{\text{trn}},n_{\text{gen}}}$ for the different models based on $n_{\text{rep}} = 100$ repetitions; see [14, Sect. 2] for how to evaluate $S_{n_{\text{trn}},n_{\text{gen}}}$. As we can see, GMMNs provide the best fit according to $S_{n_{\text{trn}},n_{\text{gen}}}$ across all dimensions considered.

Figure 2 shows (Wald-type) 95%-confidence intervals of the American basket call option price for all dependence models for different maturities (columns) and portfolios (rows) based on $n_{\text{rep}} = 25$ replications. We see that the GMMN leads to option prices that are similar to those given by the normal and the t copulas. This is not surprising as those two parametric copulas are widely used and generally believed to be "not too wrong" for these types of financial data, unlike the independence or the Clayton copulas which are "clearly wrong". However, we also notice that the pricing provided by the GMMN is often a few cents different from the ones given by the normal and t copulas. While we do not know what the true dependence model and hence what the true prices are, given that the GMMN fits better (Fig. 1) our results here suggest that, more likely than not, the normal and t copulas do not correctly capture the underlying dependence among these asset prices, either, and that those who are willing to so speculate can exploit the potential arbitrage opportunities created by the GMMN.

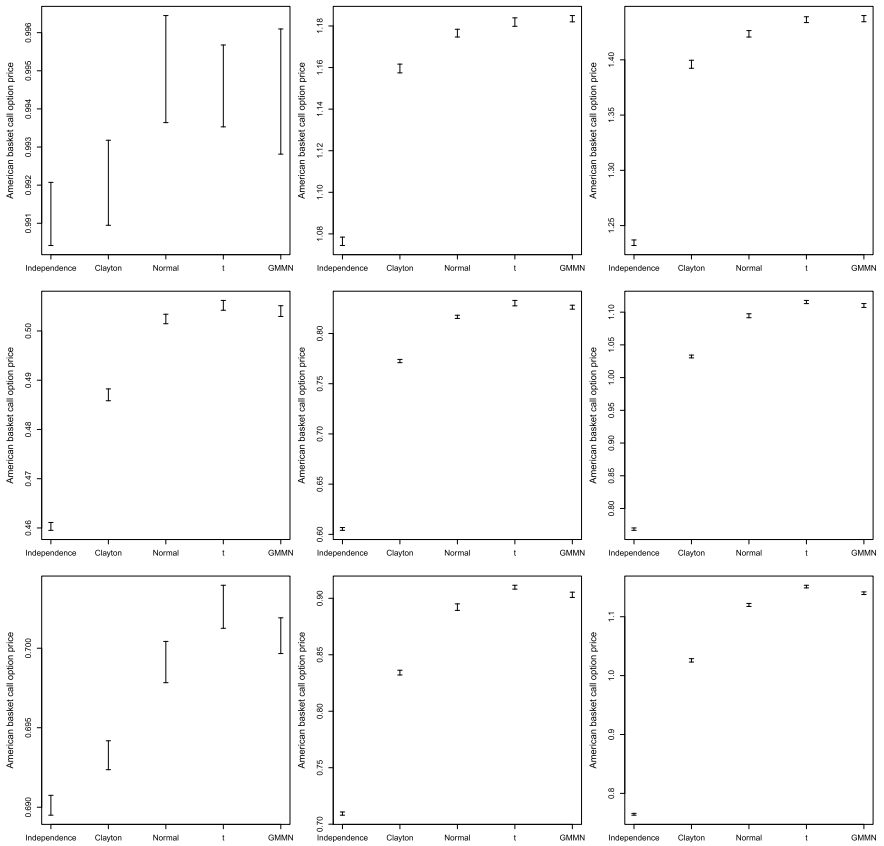


Fig. 2 95%-confidence intervals of American basket call option prices with strike prices $K = 79$ (top), $K = 69$ (middle) and $K = 81$ (bottom) based on $n_{\text{rep}} = 25$ replications and $n_{\text{pth}} = 10\,000$ sample paths in each replication at $T = 10$ (left), $T = 50$ (middle) and $T = 100$ (right) days to maturity based on portfolios of sizes $d = 3$ (top), $d = 5$ (middle) and $d = 10$ (bottom). From these plots, we see that the option prices produced by GMMNs are similar to those produced by the t and normal copulas, but still not the same as either of them

In Fig. 3 we focus on the best three models according to Fig. 1 and investigate the mean variance reduction factors (determined based on $n_{\text{rep}} = 25$ replications) when moving from pseudo-random to quasi-random numbers. We see that for shorter times to maturity we get larger variance reduction factors, but the effect deteriorates for longer times to maturity. This is true across all considered dependence models.

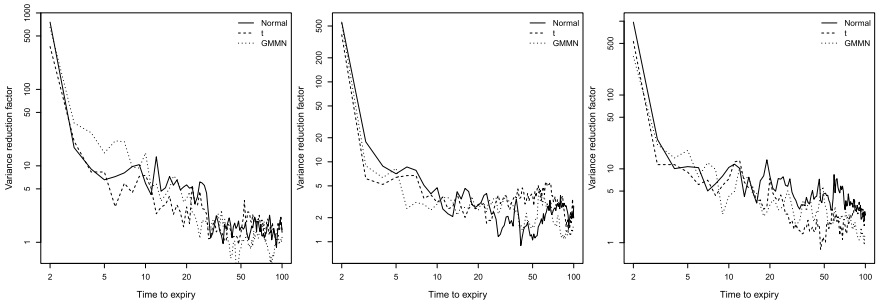


Fig. 3 Mean variance reduction factor estimates (computed over $n_{\text{rep}} = 25$ replications and $n_{\text{pth}} = 10\,000$ paths in each replication) as a function of days to maturity for portfolios of sizes $d = 3$ (left), $d = 5$ (middle) and $d = 10$ (right). From these plots, we see that the variance reduction effects are roughly similar for both the GMMNs and the two copula models and that this effect deteriorates when pricing options with longer times to maturity

4 Probabilistic Forecasting for Multivariate Time Series

In this section, we consider the application of analyzing multivariate time series data $\boldsymbol{X}_t = (X_{t,1}, \dots, X_{t,d})$ using the copula–GARCH approach [7, 13].

4.1 Model

For simplicity, we model each individual time series $X_{t,j}$ as an ARMA(1, 1)–GARCH(1, 1) process

$$\begin{aligned} X_{t_k,j} &= \mu_{t_k,j} + \sigma_{t_k,j} Z_{k,j}, \\ \mu_{t_k,j} &= \mu_j + \phi_j (X_{t_{k-1},j} - \mu_j) + \gamma_j (X_{t_{k-1},j} - \mu_{t_{k-1},j}), \\ \sigma^2_{t_k,j} &= \omega_j + \alpha_j (X_{t_{k-1},j} - \mu_{t_{k-1},j})^2 + \beta_j \sigma^2_{t_{k-1},j}, \end{aligned}$$

where $\omega_j > 0, \alpha_j, \beta_j \geq 0, \alpha_j + \beta_j < 1, |\phi_j|, |\gamma_j| < 1$, and $\phi_j + \gamma_i \neq 0$ to guarantee a causal, invertible and covariance stationary solution; see [11, Chap. 4]. Of course, higher-order ARMA–GARCH processes can also be used, but that does not affect what we are trying to demonstrate in this section.

For fixed j, the $Z_{k,j}$'s are iid according to F_{Z_j} across all time points t_k with mean zero and unit variance. In financial time series applications, it is common to model F_{Z_j} as a standard normal, standardized t or standardized skewed t distribution. In the case of the latter, it is important to note that each F_{Z_j} is allowed to have a different degree of freedom.

These marginal ARMA–GARCH models capture the serial dependence within each individual time series. To capture the cross-sectional dependence between indi-

vidual series, copulas are used to model the distribution of $(F_{Z_1}(Z_{k,1}), \ldots, F_{Z_d}(Z_{k,d}))$.

4.2 Estimation

In the context of this application, (4) amounts to the process of estimating everything—i.e., $\mu_j, \phi_j, \gamma_j, \omega_j, \alpha_j, \beta_j$ and (parameters of) F_{Z_j}—and then removing them from all the given data $X_{t_k,j}$, $k = 1, \ldots, n_{\text{trn}}$, by

$$\begin{aligned}\hat{Z}_{k,j} &= (X_{t_k,j} - \hat{\mu}_{t_k,j})/\hat{\sigma}_{t_k,j},\\ \hat{U}_{k,j} &= \hat{F}_{Z_j}(\hat{Z}_{k,j}).\end{aligned}$$

These steps are known in the literature as deGARCHing. As in Sect. 3, using the resulting $\hat{U}_{k,j}$, we can either estimate a parametric copula model, $\hat{C}_{\text{PM}}$, or train a nonparametric GMMN, $\hat{C}_{\text{NN}}$. In the case of the former, we call this the *copula–GARCH approach*; in the case of the latter, we call it the *GMMN–GARCH approach*.

4.3 Forecast

To produce probabilistic forecasts, we must now simulate each $X_{t,j}$ process forward, according to (3). Suppose we have observed the process up to and including time t_k, and would like to forecast h periods ahead; that is, we'd like to independently simulate n_{pth} paths forward and generate $\hat{X}^{(i)}_{t_{k'},j}$ for $i = 1, \ldots, n_{\text{pth}}$ and $k' = k+1, \ldots, k+h$.

For fixed i and k', this is achieved for the GMMN–GARCH approach by first generating $\boldsymbol{U}^{(i)}_{k'} = (U^{(i)}_{k',1}, \ldots, U^{(i)}_{k',d})$ with either Algorithm 1 (for pseudo-random samples) or Algorithm 2 (for quasi-random samples), then letting

$$Z^{(i)}_{k',j} = \hat{F}^{-1}_{Z_j}(U^{(i)}_{k',j}),$$

and finally simulating $\hat{X}^{(i)}_{t_{k'},j}$ according to

$$\begin{aligned}\hat{\mu}^{(i)}_{t_{k'},j} &= \hat{\mu}_j + \hat{\phi}_j(\hat{X}^{(i)}_{t_{k'-1},j} - \hat{\mu}_j) + \hat{\gamma}_j(\hat{X}^{(i)}_{t_{k'-1},j} - \hat{\mu}^{(i)}_{t_{k'-1},j}),\\ \hat{\sigma}^{2^{(i)}}_{t_{k'},j} &= \hat{\omega}_j + \hat{\alpha}_j(\hat{X}^{(i)}_{t_{k'-1},j} - \hat{\mu}^{(i)}_{t_{k'-1},j})^2 + \hat{\beta}_j\hat{\sigma}^{2^{(i)}}_{t_{k'-1},j},\\ \hat{X}^{(i)}_{t_{k'},j} &= \hat{\mu}^{(i)}_{t_{k'},j} + \hat{\sigma}^{2^{(i)}}_{t_{k'},j} Z^{(i)}_{k',j},\end{aligned}$$

where, for $k' \le k$, we simply set $\hat{X}^{(i)}_{t_{k'},j} = X_{t_{k'},j}$, $\hat{\sigma}^{2^{(i)}}_{t_{k'},j} = \hat{\sigma}^2_{t_{k'},j}$, and $\hat{\mu}^{(i)}_{t_{k'},j} = \hat{\mu}_{t_{k'},j}$ for all i. Notice that it's possible to do this at $t_k > t_{n_{\text{trn}}}$. Then, all observed quantities up to and including $\boldsymbol{X}_{t_k}$ are used to make forecasts, but we do not re-estimate anything

that has already been estimated in Sect. 4.2 using only observed quantities up to and including $\boldsymbol{X}_{t_{n_{\text{trn}}}}$.

The collection of simulated paths,

$$\{\hat{\boldsymbol{X}}^{(i)}_{t_{k+1}}, \hat{\boldsymbol{X}}^{(i)}_{t_{k+2}}, \dots, \hat{\boldsymbol{X}}^{(i)}_{t_{k+h}} \mid \mathcal{F}^{(i)}_{t_k}\}_{i=1}^{n_{\text{pth}}},$$

encode an *empirical predictive distribution* (EPD) at each time point $t_{k+1}, \dots, t_{k+h}$, from which various probabilistic forecasts can be made — for example, we can forecast $\mathbb{P}(\boldsymbol{X}_{t_{k+h}} \in A)$ by $(1/n_{\text{pth}}) \sum_{i=1}^{n_{\text{pth}}} \mathbb{1}(\hat{\boldsymbol{X}}^{(i)}_{t_{k+h}} \in A)$ for any given $A \subset \mathbb{R}^d$. However, below when we assess h-period-ahead EPDs (made at time t_k), we will only be comparing $\{\hat{\boldsymbol{X}}^{(i)}_{t_{k+h}} | \mathcal{F}^{(i)}_{t_k}\}_{i=1}^{n_{\text{pth}}}$ with $\boldsymbol{X}_{t_{k+h}}$, not any of the "intermediate" forecasts made along the way at $t_{k+1}, \dots, t_{k+h-1}$.

4.4 Application

We illustrate with two exchange rate data sets: a US dollar (USD) data set consisting of daily exchange rates of Canadian dollar (CAD), Pound sterling (GBP), Euro (EUR), Swiss Franc (CHF) and Japanese yen (JPY) with respect to the USD; and a GBP data set consisting of daily exchange rates of CAD, USD, EUR, CHF, JPY and the Chinese Yuan (CNY) with respect to the GBP. For further details regarding both the data sets, see the R package `qrmdata`.

In particular, we consider these multivariate time series from t_1=2000-01-01 to $t_{n_{\text{all}}}$=2015-12-31, treating data up to $t_{n_{\text{trn}}}$=2014-12-31 as the training set and the remainder as a held-out test set.

The distribution $F_{Z_j}(z_i) = t_{\nu_j}(z_j\sqrt{\nu_j/(\nu_j-2)})$ is chosen to be the scaled t-distribution for all $j = 1, \dots, d$. After the steps in Sect. 4.2, we fit three dependence models to $\hat{U}_{k,j}$, $k = 1, \dots, n_{\text{trn}}$, $j = 1, \dots, d$: as $\hat{C}_{\text{PM}}$, a normal copula and a t-copula, both with unstructured correlation matrices; and as $\hat{C}_{\text{NN}}$, a GMMN with the same architecture and hyperparameters as described in [6]. In addition, we use batch normalization and dropout regularization (with a dropout rate of 0.3) to help control for overfitting while training.

To assess the fit of these dependence models, we use a Cramér-von-Mises type test statistic presented in (11). From Fig. 4 which displays box plots of $S_{n_{\text{trn}},n_{\text{gen}}}$ for the different models based on $n_{\text{rep}} = 100$ repetitions, we can see that GMMNs clearly provide the best fit across both FX USD and FX GBP data sets.

The key question, though, is whether better fits translate to better predictions. For fixed horizon h, we can produce an h-day-ahead EPD $\{\hat{\boldsymbol{X}}^{(i)}_{t_{k+h}} | \mathcal{F}^{(i)}_{t_k}\}_{i=1}^{n_{\text{pth}}}$ at every $t_k = t_{n_{\text{trn}}}, \dots, t_{n_{\text{all}}-h}$ in the test period. To assess the quality of the EPDs produced by the copula–GARCH and the GMMN–GARCH approaches, we compare the respective EPD with the actual realization $\boldsymbol{X}_{t_{k+h}}$ in the held-out test set using the *variogram score* introduced by [15], which, in our context, assesses if the EPD is biased for the distance

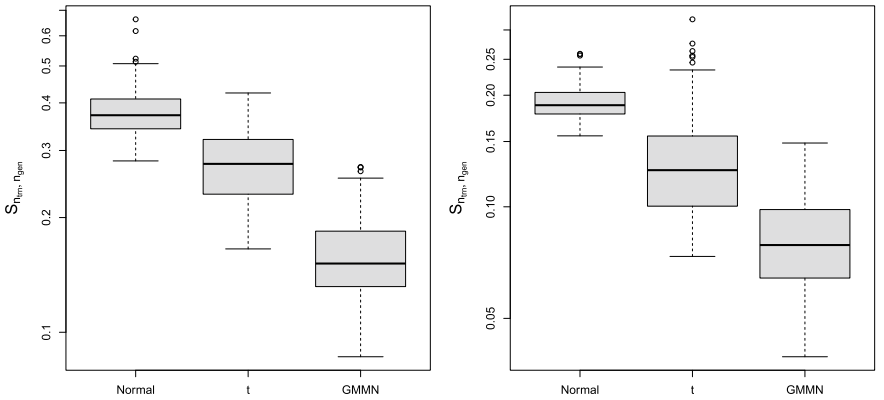


Fig. 4 Box plots based on $n_{\text{rep}} = 100$ realizations of $S_{n_{\text{trn}},n_{\text{gen}}}$ computed for the FX USD (left) and FX GBP (right) data sets of size $n_{\text{trn}} = 5478$ with dimensions $d = 5$ and $d = 6$, respectively, and for each fitted dependence model using a pseudo-random sample of size $n_{\text{gen}} = 10\,000$. From these box plots, we can see that the GMMNs provide a much better fit than the two parametric copulas for both data sets

between any two component samples. For a single numeric summary, we work with an average variogram score (of order r) over the entire period $t_{n_{\text{trn}}}, \ldots, t_{n_{\text{all}}-h}$

$$\begin{aligned}&\text{AVS}_h^r \\ &= \frac{1}{n_{\text{all}} - h - n_{\text{trn}}} \sum_{k=n_{\text{trn}}}^{n_{\text{all}}-h} \sum_{j_1=1}^{d} \sum_{j_2=1}^{d} \left(|X_{t_{k+h},j_1} - X_{t_{k+h},j_2}|^r - \frac{1}{n_{\text{pth}}} \sum_{i=1}^{n_{\text{pth}}} |\hat{X}^{(i)}_{t_{k+h},j_1} - \hat{X}^{(i)}_{t_{k+h},j_2}|^r \right)^2 .\end{aligned} \tag{12}$$

Scheuerer and Hamill [15] numerically demonstrated that, by focusing on pairwise distances between component samples, this metric discriminates well between various dependence structures. They also stated that a typical choice of the variogram order might be $r = 0.5$, but noted in their concluding remarks that smaller values of r could potentially yield more discriminative metrics when dealing with non-Gaussian data, which is why we choose to work with $r = 0.25$.

Figure 5 shows that, for both the USD and the GBP data sets, the GMMN–GARCH approach has produced better EPDs (smaller variogram scores) overall for held-out realizations in the test set. Moreover, the resulting variogram scores are also more stable (less variation) over replications of the same experiment when quasi-random (as opposed to pseudo-random) samples are used.

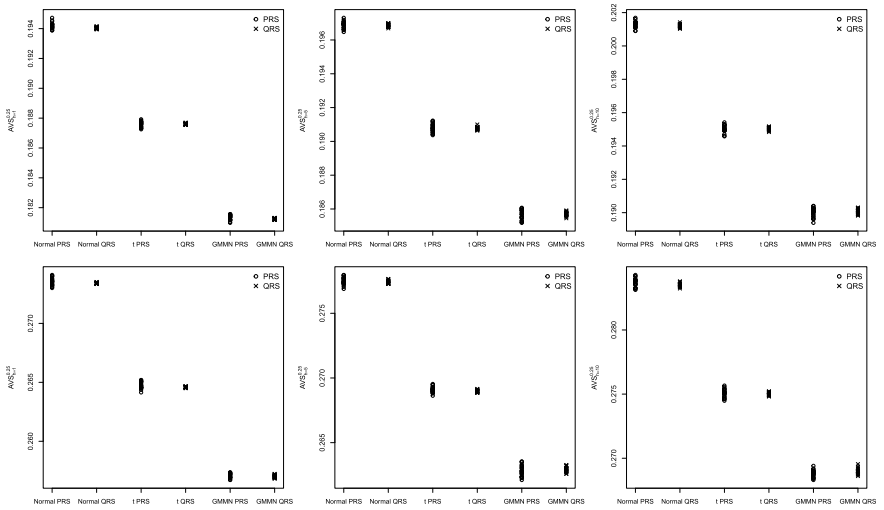


Fig. 5 Replications of $n_{\text{rep}} = 25$ average variogram scores $\text{AVS}_h^{0.25}$ based on $n_{\text{pth}} = 1000$ simulated paths in each replication, for $h = 1$ (left), $h = 5$ (middle) and $h = 10$ (right) using pseudo-random as well as quasi-random samples from normal copulas, t copulas, and GMMNs for the FX USD (top) and FX GBP (bottom) data sets. From these plots, we observe that GMMN–GARCH models yield smaller variogram scores and hence better EPDs when compared to various copula–GARCH models. Furthermore, we observe a clear variance reduction effect when using quasi-random samples to compute the variogram scores across multiple replications

5 Conclusion

We suggested GMMNs as cross-sectional dependence models for multivariate discrete-time stochastic processes. As examples, we considered discretized geometric Brownian motions with an application to pricing American basket call options under dependence, as well as ARMA–GARCH models with an application to obtain predictive distributions. These examples have demonstrated two advantages of GMMNs as dependence models. First, they provide more flexible dependence models than parametric copulas, which make a difference when estimating quantities of interest such as option prices and making probabilistic forecasts. Second, they come with a "built-in" option to generate quasi-random samples and thus allow us to obtain a variance reduction effect without additional effort.

References

1. Bollerslev, T.: Generalized autoregressive conditional heteroskedasticity **31**(3), 307–327 (1986)
2. Dziugaite, G.K., Roy, D.M., Ghahramani, Z.: Training generative neural networks via maximum mean discrepancy optimization. In: Proceedings of the Thirty-First Conference on Uncer-

tainty in Artificial Intelligence, pp. 258–267. AUAI Press (2015). http://www.auai.org/uai2015/proceedings/papers/230.pdf
3. Embrechts, P., McNeil, A.J., Straumann, D.: Correlation and dependency in risk management: Properties and pitfalls. In: Dempster, M. (ed.) Risk Management: Value at Risk and Beyond, pp. 176–223. Cambridge University Press, Cambridge (2002)
4. Genest, C., Segers, J.: On the covariance of the asymptotic empirical copula process **101**(8), 1837–1845 (2010)
5. Hofert, M., Kojadinovic, I., Maechler, M., Yan, J.: Elements of Copula Modeling with R. Springer Use R! Series (2018). https://doi.org/10.1007/978-3-319-89635-9, http://www.springer.com/de/book/9783319896342
6. Hofert, M., Prasad, A., Zhu, M.: Quasi-random sampling for multivariate distributions via generative neural networks, pp. 1–24 (2021)
7. Jondeau, E., Rockinger, M.: The copula-GARCH model of conditional dependencies: An international stock market application **25**, 827–853 (2006)
8. Lemieux, C.: Monte Carlo and Quasi–Monte Carlo Sampling. Springer, Berlin (2009)
9. Li, Y., Swersky, K., Zemel, R.: Generative moment matching networks. In: International Conference on Machine Learning, pp. 1718–1727 (2015)
10. Longstaff, F.A., Schwartz, E.S.: Valuing american options by simulation: a simple least-squares approach **14**(1), 113–147 (2001)
11. McNeil, A.J., Frey, R., Embrechts, P.: Quantitative Risk Management: Concepts, Techniques, Tools, 2 edn. Princeton University Press (2015)
12. Nelsen, R.B.: An Introduction to Copulas. Springer, Berlin (2006)
13. Patton, A.J.: Modelling asymmetric exchange rate dependence **47**(2), 527–556 (2006). http://public.econ.duke.edu/~ap172/Patton_IER_2006.pdf
14. Rémillard, B., Scaillet, O.: Testing for equality between two copulas **100**(3), 377–386 (2009)
15. Scheuerer, M., Hamill, T.M.: Variogram-based proper scoring rules for probabilistic forecasts of multivariate quantities **143**(4), 1321–1334 (2015)
16. Sklar, A.: Fonctions de répartition à n dimensions et leurs marges **8**, 229–231 (1959)
17. Weiss, A.: Arma models with arch errors **5**(2), 129–143 (1984)

Artificial Neural Networks Generated by Low Discrepancy Sequences

Alexander Keller and Matthijs Van keirsbilck

Abstract Artificial neural networks can be represented by paths. Generated as random walks on a dense network graph, we find that the resulting sparse networks allow for deterministic initialization and even weights with fixed sign. Such networks can be trained sparse from scratch, avoiding the expensive procedure of training a dense network and compressing it afterwards. Although sparse, weights are accessed as contiguous blocks of memory. In addition, enumerating the paths using deterministic low discrepancy sequences, for example variants of the Sobol' sequence, amounts to connecting the layers of neural units by progressive permutations, which naturally avoids bank conflicts in parallel computer hardware. We demonstrate that the artificial neural networks generated by low discrepancy sequences can achieve an accuracy within reach of their dense counterparts at a much lower computational complexity.

Keywords Neural networks · Low discrepancy sequences · Sparsity · Deterministic algorithms · Parallel computer hardware

1 Introduction

The average human brain has about 10^{11} nerve cells, where each of them may be connected to up to 10^4 others. Yet, the complexity of artificial neural networks quite often is determined by fully connected sets of neurons. Therefore, we investigate algorithms for artificial neural networks that are linear in the number of neurons and explore their massively parallel implementation in hardware.

In order to reduce complexity, we motivate the principle of representing an artificial neural network by paths instead of matrices in Sect. 2. Rather than creating such sparse networks by importance sampling paths from a trained dense network, training may be much more efficient when considering artificial neural networks that

A. Keller (✉) · M. Van keirsbilck
NVIDIA, Fasanenstr. 81, 10623 Berlin, Germany
e-mail: akeller@nvidia.com

M. Van keirsbilck
e-mail: matthijsv@nvidia.com

A. Keller (ed.), *Monte Carlo and Quasi-Monte Carlo Methods*, Springer Proceedings in Mathematics & Statistics 387, https://doi.org/10.1007/978-3-030-98319-2_15

are sparse from scratch as discussed in Sect. 3. Enumerating the paths of an artificial neural network using a deterministic low discrepancy sequence and exploiting its structural properties, leads to an efficient hardware implementation, whose advantages are detailed in Sect. 4. Initial numerical evidence to support the approach is reported in Sect. 5 before drawing the conclusions. A noteworthy result is that quasi-Monte Carlo methods enable a completely deterministic approach to artificial neural networks.

2 Representing Artificial Neural Networks by Paths

In order to provide an intuition why representing artificial neural networks as paths may lower their computational complexity, we review their basic principles.

As depicted in Fig. 1, the computational graph of a basic artificial neural network or multi-layer perceptron (MLP) may be organized in $L+1 \in \mathbb{N}$ layers, each comprising of $n_l \in \mathbb{N}$ neural units, where $0 \leq l \leq L$. Given an input vector a_0, the output vector a_L is computed layer by layer, where each vertex determines the activations

$$a_{l,i} := \max\left\{0, \underbrace{\sum_{j=0}^{n_{l-1}-1} w_{l,j,i} \cdot a_{l-1,j}}_{=:z_{l,i}}\right\}. \tag{1}$$

For the purpose of the article, it is sufficient to consider the non-linearity $\max\{0, x\}$ as an activation function, yielding the so-called rectified linear unit (ReLU). The vertices of the graph are connected by edges with their associated weights $w_{l,j,i} \in \mathbb{R}$. In summary, each neural unit computes a weighted average of the activations in the

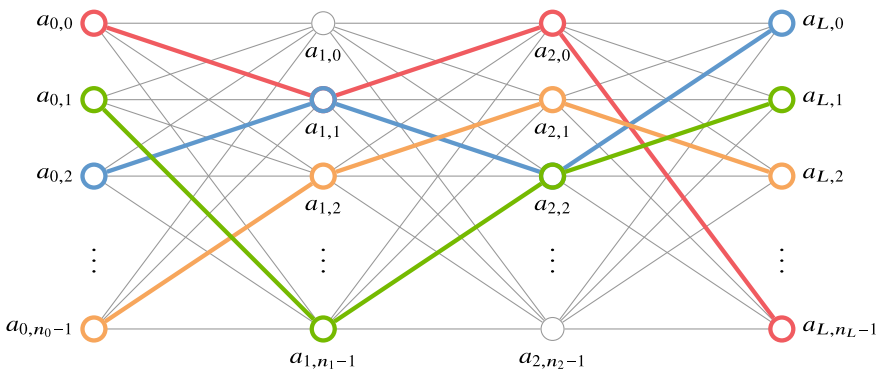


Fig. 1 Representing the graph of an artificial neural network by paths (colored) instead of fully connected layers (including gray) allows for algorithms linear in the number of vertices in time and space

previous layer. If the average is non-positive, it is clipped to zero, which renders the neural unit inactive. Otherwise the neural unit is called active and passes on the positive average.

In order to learn the weights from training data, backpropagation [35] has become the most popular algorithm: Given an input vector a_0 and a desired output vector d, the approximation error

$$\delta_L := a_L - d \tag{2}$$

is propagated back through the network by computing the weighted average of the error

$$\delta_{l-1,i} := \sum_{a_{l,j}>0} \delta_{l,j} \cdot w_{l,j,i} \tag{3}$$

of all active neural units in a layer. If a neural unit is active, the weight

$$w'_{l,j,i} := w_{l,j,i} - \lambda \cdot \delta_{l,j} \cdot a_{l-1,i} \text{ if } a_{l,j} > 0 \tag{4}$$

of an edge connecting it to a previous neural unit is updated by the product of the learning rate $\lambda \in \mathbb{R}^+$, the error at the active neural unit, and the activation of the previous neural unit.

As formalized by Eq. 1 and shown in Fig. 1, all neural units of one layer are connected to all neural units of the next layer. Such "fully connected" layers are found in many modern artificial neural networks, for example at the end of classification networks or as so-called 1x1-convolutions, which are fully connected layers with weight sharing across inputs. Obviously, the computational complexity as well as the number of weights of a layer is determined by the product of the number of neural units in the current and previous layer.

In order to motivate an algorithm linear in time and space, we rewrite Eq. 1, which is the non-linearity applied to the average, equivalently as average of the non-linear activation functions:

$$z_{l,i} = \sum_{j=0}^{n_{l-1}-1} w_{l,j,i} \cdot \max\{0, z_{l-1,j}\}$$

Considering an integral

$$z_l(y) := \int_0^1 w_l(x, y) \cdot \max\{0, z_{l-1}(x)\} dx$$

rather than a sum, reveals that layers in artificial neural networks relate to high-dimensional integro-approximation. Hence, in continuous form, an artificial neural network is a sequence of linear integral operators applied to non-linear functions.

From the domain of integral equations, especially the domain of computer graphics, we know how to deal with such sequences: Sampling path space, we trace light

transport paths that connect the light sources and camera sensors to render synthetic images. It now is obvious that an artificial neural network may be represented by paths that connect inputs and outputs, too. Computation only along the paths (colored in Fig. 1) results in a complexity in space and time that is linear in the number of paths times the depth of the neural networks and hence may be linear in the number of neural units.

2.1 *Quantization of Artificial Neural Networks by Sampling*

In [29] we derived an algorithm that quantizes a trained artificial neural network such that the resulting complexity may be linear. To create these paths given a trained neural network, we exploit an invariant of the rectified linear units (ReLU): In fact, scaling the activations by a positive factor $f \in \mathbb{R}^+$ and dividing the weighted average by the same factor leaves the result unchanged.

Choosing this factor as the one-norm of the weights of a neural unit, the factor can be propagated forward through the neural network, leaving the weights of each neural unit as a discrete probability density.

Given the n weights of a single neural unit, assuming $\sum_{k=0}^{n-1} |w_k| = \|w\|_1 = 1$ and defining a partition of the unit interval by $P_m := \sum_{k=1}^{m} |w_k|$, it is straightforward to trace paths from the outputs back to the inputs by sampling proportional to the discrete densities. The graphs in Fig. 2 provide evidence that sampling only a fraction of the connections results in no notable degradation in the test accuracy. A similar approach can be used to quantize not just weights, but also activations and gradients to arbitrary precision. For more details and data, we refer to [29, 30].

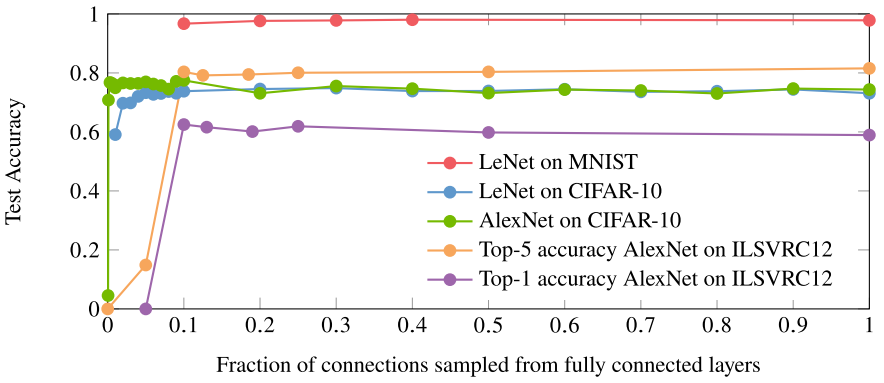


Fig. 2 Test accuracy of a selection of classic artificial neural networks for image recognition. Using only about 10% (or even less) of the connections of the original trained networks does not result in a notable loss in test accuracy, indicating potential efficiency gains. The paths through the fully connected layers of the artificial neural networks have been sampled proportionally to the trained weights

Above we used the L^1-norm to generate probabilities from the weights. It is also possible to use more advanced importance estimation techniques. This can be used both during training to precondition the model so it can be pruned to higher sparsity levels [28]. Similarly, subsets of neurons or weights can be selected such that a certain sparsity level is maintained throughout training for increased efficiency [3, 17].

2.2 Sampling Paths in Convolutional Neural Networks (CNNs)

Convolutional neural networks [27] contain layers that compute features by convolutions. For example, common features include first and second derivatives to identify edges of different orientations in an image.

A convolutional neuron (also called filter or kernel) is specified by a 3D tensor of weights of width w, height h, and depth c_{in} (also called channel dimension). Typically, the dimensions $w \times h$ are small, for example, 3×3. Given an input tensor of width W, height H, and depth c_{in}, each 2D depth slice is convolved with the corresponding 2D depth slice of shape $w \times h$ of the weight tensor. Then the resulting features are summed along the depth dimension to produce one output feature channel of shape $W \times H \times 1$. With c_{out} convolutional neurons each computing one output channel from the input tensor, a CNN layer may be interpreted as a function that maps c_{in} input channels to c_{out} output channels, just like an MLP.

Hence, to create a sparse CNN, we trace an edge of a path the same way as for MLPs: by selecting one of the c_{in} input channels, and one of the c_{out} convolutional neurons in the layer. This activated edge means the selected $w \times h$ depth slice in the 3D weight tensor of the selected neuron will be convolved with the corresponding slice of the input tensor.

Many recent CNN architectures such as MobileNet, DenseNet, or QuartzNet [23] use 1×1-convolutions, where $w = h = 1$. This important special case amounts to a structure identical to a fully connected layer [1, 15], where weights are reused across the elements of the input tensor. Paths are traced in the same way as described before.

Tracing paths through trained convolutional networks is related to "channel pruning" [28]. This enables coarse sparsity on the filter level, which is more efficient on current hardware than fine-grained sparsity [1], i.e. selecting single weights.

3 Training an Artificial Neural Network Sparse from Scratch

Quantizing an artificial neural network by sampling paths still requires to train the full network. This complexity issue may be resolved by training an artificial neural network sparse from scratch as principled by the implementation provided in Fig. 3.

Inference with an Artificial Neural Network represented by Paths

```
// initialization

int neurons = 0;

for (int l = 0; l < layers; ++l)
{
  for (int p = 0; p < paths; ++p)
    index[l][p] = neurons + (int) (drand48() * neuronsPerLayer[l]);

  neurons += neuronsPerLayer[l];
}

for (int l = 1; l < layers; ++l)
  for (int p = 0; p < paths; ++p)
    weight[l][p] = initialWeight; // deterministic instead of random

float *a = new float[neurons];
float *error = new float[neurons];

// train by backpropagation

...

// inference

for (int i = 0; i < neuronsPerLayer[0]; ++i)
  a[i] = inputs[i];

for (int i = neuronsPerLayer[0]; i < neurons; ++i)
  a[i] = 0.0f; // or bias[i], if bias terms are used

for (int l = 1; l < layers; ++l)
  for (int p = 0; p < paths; ++p)
    if (a[index[l - 1][p]] > 0.0f)  // ReLU
       a[index[l][p]] += weight[l][p] * a[index[l - 1, p]];
```

Fig. 3 Implementation of an artificial neural network represented by paths using rectified linear units (ReLU) as activation functions. Given the numbers of `layers`, `paths`, and the array of `neuronsPerLayer`, the array `index` stores the indices of neural units along a path `p` created by randomly selecting a neural unit per layer. Before training by backpropagation, the `weight` of each edge is set to a constant `initialWeight`. For inference, the activations `a` of the first layer are set to the `input` data, while the remaining activations are set to zero. Enumerating all activations for all subsequent layers and for all paths, each activation along an edge is updated, if its previous activation along the path is active, i.e. larger than zero, which amounts to the rectified linear unit (ReLU) activation function

We start by storing random paths as an array `index[][]` of indices and enumerate all layers and paths. In layer l of path p, the index of a neural unit is just randomly selected among the neuron units of that respective layer.

The evaluation—also called inference—of an artificial neural network represented by paths first copies all inputs to an array of activations and then initializes all other activations to zero. The actual computation then loops over all layers and paths, where an activation is updated only if the previous vertex along the path p is positive, meaning active. This is an implicit implementation of the rectified linear neural unit (ReLU) introduced in Eq. 1.

In the same manner, restricting training by backpropagation [35] as defined by Eqs. 2, 3, and 4 to the representation by paths is straightforward. In analogy to Sect. 2.2, convolutional layers can be represented by paths and be trained sparse from scratch, too, resembling methods to create predefined sparse kernels [24, 25].

It is obvious that the complexity of inference and training is linear in the number of paths times the depth of the neural network. Also note, that although sparse, all weights are accessed in linear order, which is the most efficient memory access pattern on modern computer hardware.

For random paths, multiple paths may select the same weights while leaving others untouched. This wastes memory and computation and may make training more difficult. A solution to this issue is to use low discrepancy sequences to generate the paths, as will be discussed in Sect. 4.

3.1 Constant Initialization

For a fully connected neural network, all neurons in a layer have the same connectivity, i.e. each neuron is connected to the same neurons in the previous and following layer. If all weights were initialized uniformly, all neurons would receive the same updates, and the network would learn nothing during training. The usual way to prevent this is to initialize the weights by sampling randomly from some distribution. There has been a lot of work done on finding good initializations, depending on the used activation function, size of weight tensor, and other factors [11, 13, 37].

However with sparse networks, each neuron has a different connectivity pattern. Instead of introducing randomness by sampling random weights, the non-uniform connectivity pattern ensures that not every neuron learns the same thing. This allows one to get rid of the random initialization, as shown in the code in Fig. 3, where the weights along the edges of the paths are initialized with a constant.

The value of the constant itself is still important as it controls the operator norm of the affine transformation that each neuron performs. Following the analysis of [13], and considering that our networks use the ReLU activation function, we use $w_{\text{init}} = \frac{6}{\sqrt{\texttt{fan\_in}+\texttt{fan\_out}}}$, where `fan_in` is the number of inputs to a neural unit and `fan_out` the number of its connections to the next layer. Biases are initialized

with 0, and scale and shift parameters of batch normalization layers are initialized with 1 and 0, respectively.

We may conclude that the classic random weight initialization required to make fully connected layers learn is replaced by the fact that in a artificial neural network sparse from scratch neural units don't share the same set of connections.

3.2 Non-negative Weights

Weights may change sign during training. Yet, it has been observed that a graph of an artificial neural network including static signs of the weights may be separated from training the magnitudes of the weights [10, 40]. However, finding this graph and its associated static signs requires pruning a trained, fully connected, and randomly initialized artificial neural network.

When working with optical implementations of artificial neural networks, only non-negative weights may be used, because either there is light or not [9]. The lack of negative weights is accounted for by amplifying differences of weighted sums similar to how operational amplifiers work. A modern example are ternary quantized artificial neural networks [41], where a first binary matrix accounts for all non-negative weights of a layer, and a second binary matrix produces the sums to be subtracted. Still, finding the ternary representation requires pruning a fully connected network and retraining.

Representing artificial neural networks by paths, we propose to attach one fixed sign to each path. As paths are generated by random walks, selecting the weights of the even paths to be non-negative and the weights of the odd paths to be non-positive perfectly balances the number of positive and negative weights. Alternatively, any ratio of positive and negative weights may be realized by for example determining the sign of a path by comparing its index to the desired number of positive paths (even per layer). This architecture can be thought of as an inhibiting network superimposed on a supporting network. Such a network may be trained by, for example, backpropagation with the only restriction that weights cannot become negative.

3.3 Normalization

Using a constant value for weight initialization allows one to fulfill normalization constraints. For example, knowing the number of edges incident to a neural unit, it is straightforward to determine the initial weight such that any selected p-norm of a set of weights will be one. Uniformly scaling the initial weights allows one to control the operator norm of the artificial neural network represented by paths [29].

4 Low Discrepancy Sequences to enumerate Network Graphs

Low discrepancy sequences [8, 31] may be considered the deterministic counterpart of pseudo-random number generators. Abandoning the simulation of independence, they generate points in the unit hypercube much more uniformly distributed than random numbers ever can be. Improving convergence speed, they have become the industry standard for generating light transport paths in computer graphics [14, 21].

Reviewing classic concepts of parallel computation (see Sect. 4.1), taking advantage of the properties of low discrepancy sequences (see Sect. 4.2) yields an algorithm to enumerate the graph of an artificial neural network (see Sect. 4.3) that perfectly suits an implementation in massively parallel hardware (see Sect. 4.4).

4.1 Parallel Computer Architecture

Already in the 1970s, the concepts of systolic arrays [26] and the perfect shuffle [38] have been investigated in the context of parallel computer architecture.

Systolic arrays are based on simple processing units that are chained to form pipelines. As an example, the top of Fig. 4 shows a multiply-and-add-unit that computes $a \cdot x + b$. Both x and b are buffered by a register. Chaining multiple such compute units allows one to parallelize large parts of matrix multiplication. Obviously the latency of a systolic array pipeline is determined by the length of the chain of processing elements. The notion "systolic" stems from the analogy to the heart pumping in and out blood within a heartbeat, where in systolic arrays data is pumped

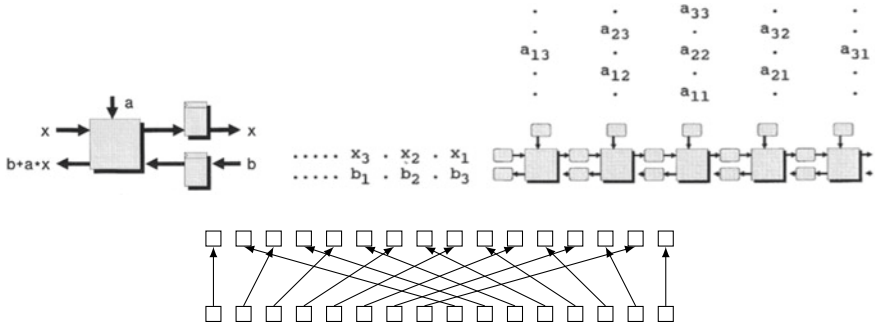


Fig. 4 Parallel computer architecture. Top: Systolic arrays tile identical processing units for parallel processing. Here, instances of a multiply-and-add unit with input and output registers are chained to parallelize matrix multiplications. Bottom: Linking registers and processing units using the interleaving permutation as given by a perfect shuffle allows for parallelizing many useful computations, the most prominent example being the fast Fourier transform (FFT)

in and out within a cycle. In fact, Google's tensor processing units (TPU) are based on this architecture.

Perfect shuffle networks connect an array of registers to an array of processing units by the permutation resulting from perfectly interleaving two decks of cards one by one from each deck (see the bottom part of Fig. 4). The architecture has many famous applications, including the efficient implementation of the fast Fourier transform (FFT). The number of iterations is the latency, which amounts to the logarithm in base 2 of the number of registers, i.e. values to process.

Unrolling the iteration results in a structure reminiscent of the layer structure of artificial neural networks (see Fig. 1) and in fact has been tried to construct simple optical neural networks [34]. Yet, the connection pattern of the perfect shuffle appears to be too restrictive. In a series of articles [4–6] more general permutations to connect layers have been explored. The permutations have their origins in interleaver design and interleaved codes. Visualizing the connection patterns in the unit square [4, Figs. 4 and 5], where a point means a connection of the neuron at coordinate x to a neuron in the next layer at coordinate y in the subsequent layer, tends to make one think of sampling patterns as used in random number generation and quasi-Monte Carlo methods [31].

4.2 Progressive Permutations

Many low discrepancy sequences are based on radical inversion. The principle is best explained by taking a look at the van der Corput sequence

$$\begin{aligned} \Phi_b : \mathbb{N}_0 &\to \mathbb{Q} \cap [0, 1) \\ i = \sum_{l=0}^{\infty} a_l(i) b^l &\mapsto \Phi_b(i) := \sum_{l=0}^{\infty} a_l(i) b^{-l-1} \end{aligned}$$

in base $b \in \mathbb{N} \setminus \{1\}$ that maps the integers to the unit interval: Representing the integer i as digits $a_l(i)$ in base b and mirroring this representation at the decimal point yields a fraction between zero and one.

For contiguous blocks of indices $k \cdot b^m \leq i < (k+1) \cdot b^m - 1$ for any $k \in \mathbb{N}_0$, the radical inverses $\Phi_b(i)$ are equidistantly spaced. As a consequence of this perfect stratification, the integers $\lfloor b^m \Phi_b(i) \rfloor$ are a permutation of $\{0, \ldots, b^m - 1\}$. Fixing b^m, k enumerates a sequence of permutations. As an example for $b = 2$, the first $2^4 = 16$ points yield the permutation

$$16 \cdot \Phi_2(i)|_{i=0}^{15} = (0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15).$$

These properties are shared by the individual components of the s-dimensional Sobol' sequence [36], which may be the most popular low discrepancy sequence: Its first component is $x_i^{(0)} := \Phi_2(i)$, while the subsequent components

$$x_i^{(j)} = (2^{-1} \cdots 2^{-m}) \cdot \underbrace{\left(C_j \cdot \begin{pmatrix} a_0(i) \\ \vdots \\ a_{m-1}(i) \end{pmatrix} \right)}_{\text{in } \mathbb{F}_2} \in \mathbb{Q} \cap [0, 1) \tag{5}$$

multiply a generator matrix C_j with the vector of digits before radical inversion. The generator matrices C_j are determined by the j-th primitive polynomial and for more details we refer to [8, 18, 19, 36].

The matrix vector multiplication takes place in the field $\mathbb{F}_2$ of two elements and very efficiently can be implemented using bit-wise parallel operations on unsigned integers. For each digit set in the integer i, the corresponding column of the generator vector just needs to be xor-ed with the so far accumulated value:

```
unsigned int x = 0;

for (unsigned int k = 0; i; i >>= 1, ++k)
  if (i & 1)
       x ^= C[k]; // parallel addition of column k of the matrix C
```

Experimenting with the Sobol' sequence [36] is very practical, because an efficient implementation [18] along with the source code and generator matrices has been provided at https://web.maths.unsw.edu.au/~fkuo/sobol/. Blocking groups of bits during radical inversion allows for an even faster generation of the Sobol' sequence, see [39, Listing 3.2].

The permutation properties described above are a consequence of each component of the Sobol' sequence being a $(0, 1)$-sequence in base $b = 2$. As the Sobol' sequence produces Latin hypercube samples for each number of points being a power of 2, it can be used to create permutations in a progressive way. For more detail on (t, s)-sequences that in fact are sequences of (t, m, s)-nets, we refer to [31, Chap. 4].

4.3 Sampling Quasi-Random Paths

We will use the components of the Sobol' sequence instead of the pseudo-random number generator that sampled the path indices of the sparse networks in Sect. 3. As contiguous blocks of lengths of powers of 2 form permutations, we choose a power of 2 neurons per layer. This links the neural units

$$\left(a_{l, \left\lfloor n_l \cdot x_i^{(l)} \right\rfloor}, a_{l+1, \left\lfloor n_{l+1} \cdot x_i^{(l+1)} \right\rfloor} \right) \tag{6}$$

along the i-th path according to Eq. 5. Similar to generating the paths by a pseudo-random number generator, the connectivity of the network does not need to be stored explicitly, because the components of the Sobol' sequence in Eq. 5 can be computed

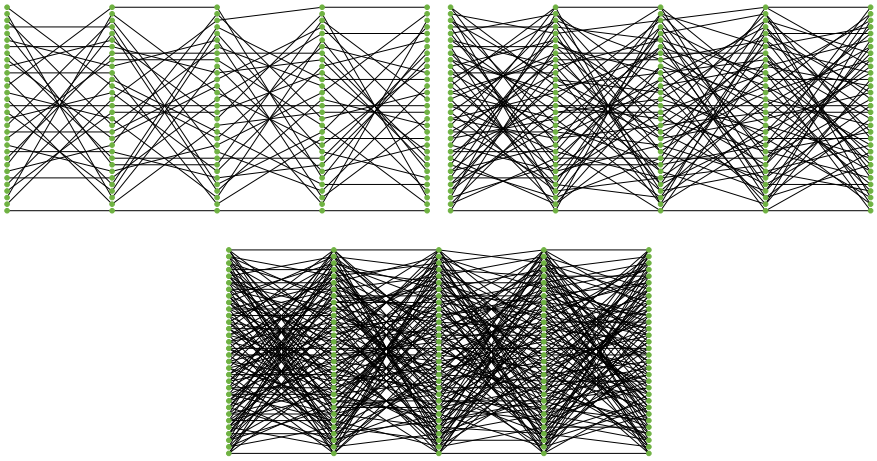


Fig. 5 Progressive enumeration of paths: For each 32 neural units in 5 layers, 32 (top left), 64 (top right), and 128 (bottom) paths generated by the Sobol' sequence are shown. The number of paths per neural unit is 1, 2, or 4, respectively

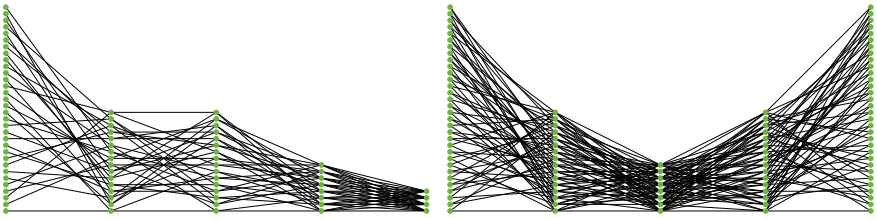


Fig. 6 Illustration of classic network architectures generated by the Sobol' sequence. Left: 32 inputs are encoded to 4 outputs. Such architectures are typically used for classification tasks. Right: 32 inputs are encoded to a latent space of 8 neural units and decoded back to 32 outputs. This is a common architecture of auto-encoders, whose typical task is filtering signals. Note that the number of neural units in each layer and the number of paths are powers of 2 and the fan-in and fan-out is constant across each layer

on the fly. When the numbers of neurons in the input or output layer are not powers of two, one may choose to just fully connect these layers with their corresponding hidden layers.

The example in Fig. 5 demonstrates one advantage of encoding the network topology by a low discrepancy sequence. As the permutations are progressive, it is straightforward to add another power of 2 connections. Enumerating the network topology from sparse to fully connected becomes natural.

Figure 6 shows an example of a sparse classifier network generated by the Sobol' sequence. A high dimensional input vector on the left is condensed to a vector of classes on the right. Each layer has a power of 2 neurons and each neuron in a layer has the exact same constant number of connections. The next example is an

autoencoder structure that is often used for filtering signals. Input and output layer are of the same dimension.

Creating a sparse network with non-negative weights like in Sect. 3.2 is as simple as selecting the first half of the paths to have non-negative weights and the second half to have non-positive weights. A second option is to dedicate one dimension of the Sobol' sequence to determine whether the weights of a path shall be non-negative or non-positive just by checking whether the component is smaller than $\frac{1}{2}$ or not. More details on partitioning one low discrepancy sequence into many are found in [22]. If the number of paths is a power of 2, partitioning a network generated by the Sobol' sequence into supporting and inhibiting network as described will result in a zero sum of weights per neuron if neurons in a layer have constant valence. This nicely complements the normalized initialization (see Sect. 3.1 and Sect. 3.3) and typically is not guaranteed when using a pseudo-random number generator to generate the paths.

When the number of paths exceeds the product of the number of neurons in two successive layers, edges will be selected more than once. Even before reaching that bound and although the components of the Sobol' sequence create progressive permutations, it may happen that multiple edges as defined by Eq. 6 coincide. While coalescing edges are not a problem for the algorithm in Fig. 3, having multiple weights associated to one edge is redundant and may reduce the capacity of the network. This reason for this issue has been known for a long time in the domain of quasi-Monte Carlo methods and especially from the Sobol' sequence, where low dimensional projections may expose very regular correlation patterns between the dimensions. To improve on the issue, low discrepancy sequences have been scrambled [32] and optimized [18, 20]. We have been successful by simply omitting the dimensions of the Sobol' sequence whose generator matrices result in coalescing edges, which can be interpreted as a permutation of the sequence of the Sobol' generator matrices. Using the cascaded construction of Sobol' points [33] that forces each successive pair of dimensions to form a $(0, m, 2)$-net, coalescing edges are avoided by construction. The respective numerical results are presented in Sect. 5.3.

4.4 Hardware Considerations

Representing sparse networks by paths, the algorithm in Fig. 3 linearly streams the weights from memory. Such an access pattern perfectly matches the parallel loading of contiguous blocks of weights in one cache line by the pre-fetcher as it is common for current processor hardware.

Similar to [7] and [6, Fig. 4], the permutations generated by the Sobol' sequence guarantee streaming weights in contiguous blocks of size of a power of 2 free of memory bank conflicts. For the same reason, weights can be routed without collisions through a crossbar switch inside the processor. Both advantages cannot be guaranteed when creating paths by pseudo-random number generators.

Determining the permutations generated by Φ_2 in hardware amounts to bit reversal, which is straightforward to hardwire. Implementing the permutations generated by Eq. 5 in hardware requires to unroll the loop over the bit-parallel XOR operations (see the algorithm in Sect. 4.2). This results in a tiny circuit with a matrix of flip-flops to hold the generator matrix C_j. Replicating parts of the circuit for all numbers representable by the m least significant bits allows one to create 2^m values of the permutation in parallel.

For backpropagation, we can take advantage of the fact the Sobol' sequence is invertible. Computing the inverse of Eq. 5 just requires to determine C_j^{-1}. Propagating errors back through the network, the memory access remains contiguous when enumerating the array of weights backwards.

5 Numerical Results

We perform numerical experiments to evaluate the accuracy of neural networks represented by paths generated by the Sobol' low discrepancy sequence. We take a look at classification tasks using classic multilayer perceptrons (MLP) as illustrated in Fig. 1, convolutional neural networks, the set of hyperparameters, and the initialization of sparse neural networks.

5.1 Performance of Sparse Neural Networks Represented by Paths

Using the algorithm in Fig. 3, we demonstrate the linear complexity of sparse neural networks represented by paths as compared to fully connected networks represented by matrices. While it is possible to compress sparse matrices, there is an additional cost for decompression that needs to be amortized. In contrast, representing the neural network by paths, a linear speedup results for any desired sparsity. Figure 7 compares the relative runtimes for 1 epoch for a network of four layers with 256 neurons per layer, varying the number of paths, and compares them to the sparsity and accuracy. The experiments were run on a single core of an AMD 7 5800X CPU.

5.2 Training Sparse from Scratch

For the simple examples of recognizing digits in tiny images (see Fig. 7), training a sparse-from-scratch artificial neural network reveals that only a tiny fraction of paths as compared to the number of connections within the fully connected network is required to come very close to the test accuracy of the fully connected variant.

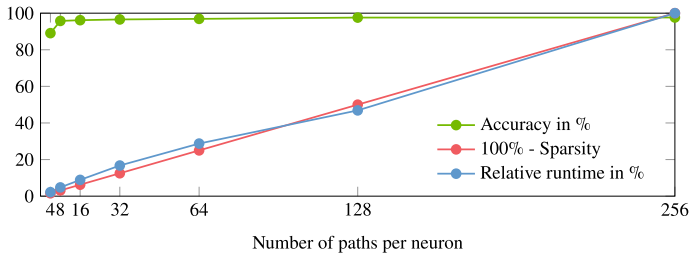


Fig. 7 Accuracy, sparsity and runtime relative to the fully connected neural network for image classification on MNIST for sparse networks represented by paths with 4 hidden layers of 256 neurons

Comparing paths generated by pseudo-random numbers to paths generated by a low discrepancy sequence does not show a big difference in accuracy compared to random walks. However, as stated in Sect. 4.3, using the Sobol' low discrepancy sequence allows for routing without bank conflicts and avoids duplicate weights. This guarantee is also a big advantage over pseudo-random number generated access patterns when considering a hardware implementation [4, 6].

5.3 Random and Quasi-Random Paths in CNNs

Similar to the fully connected neural networks, convolutional neural networks (CNNs) represented by paths (see Sect. 2.2) can be trained sparse from scratch. For the numerical experiments, we use the CIFAR-10 image recognition data set. Our CNN has 5 convolutional layers with a number of channels 16, 32, 32, 64, 64, respectively, followed by one fully connected layer with a softmax activation function to produce 10 output features each identifying one of the dataset classes. Every convolutional layer is followed by a Batch Norm layer [16] and a ReLU activation function [11]. Training is done using stochastic gradient descent (SGD) with a momentum of 0.9 and weight decay of 0.0001, for 182 epochs. The learning rate starts at 0.1, and is decreased by factor 10 at epochs 91 and 136. We normalize the input images using the mean and standard deviation over the training set, and apply additional augmentation in the form of random horizontal flips as well as a 32×32 crop after padding the input image 4 pixels on every side.

Figure 8 shows the accuracy of sparse from scratch CNNs compared to a fully connected CNN, and Fig. 9 shows the corresponding number of non-zero parameters. We observe a sharp increase in accuracy initially and then a slower convergence towards the accuracy of the fully connected network. Importantly, the graphs show that an accuracy close to the fully connected network can be reached with far fewer weights by sparse networks.The figure also shows that sampling paths randomly or quasi-randomly using the Sobol' sequence performs very similarly in terms of

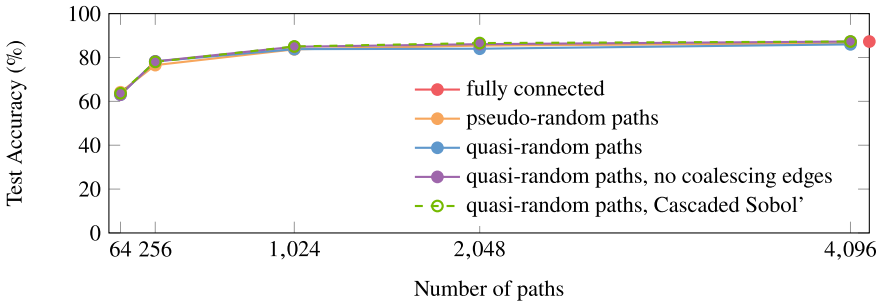


Fig. 8 Test accuracy of a convolutional neural network (CNN) represented by paths and trained sparse from scratch compared to the fully connected counterpart. The task is recognizing 10 classes of objects in 32 × 32 pixel images (CIFAR-10). Around 1024 paths the accuracy reaches a plateau very close to the maximum accuracy, advocating sparse networks

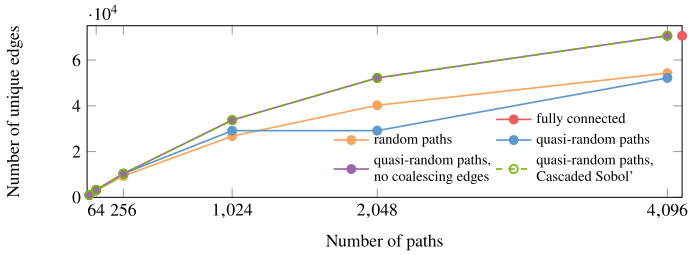


Fig. 9 Sparse networks created with random paths may have some parts of the paths overlap, creating coalescing edges. For the Sobol' sequence it may occur that certain combinations of dimensions result in coalescing edges as well (blue line). This can be resolved by simply omitting the dimensions of concern (purple line). Cascaded Sobol' points avoid coalescing edges by construction. Note that for 1024 paths the accuracy (see Fig. 8) already has reached a plateau

accuracy. However, as remarked in Sect. 4.3, there are large potential hardware advantages when using quasi-random methods.

5.4 Relation of Number of Paths, Layer Width, and Accuracy

It has been observed that wider but sparse networks have higher representational capacity than narrower dense networks [2, 12]. Selecting the convolutional neural network as in the previous section, we try to verify this empirically in Table 1, where we investigate how the accuracy changes when we scale the number of neurons per layer, i.e. the network width, but keep the number of weights constant. This can be done by increasing the sparsity as the width increases. In a fully connected network the number of weights increases quadratically with width, whereas for networks defined by paths the number of weights is determined by the number of paths. The

Table 1 Comparing fully connected to wider, sparser networks created by random paths. The number of paths is chosen such that all networks have around 70400 weights like the fully connected network. All networks are close, but highest accuracy is achieved for width multiplier 1.25

Width multiplier	Number of paths	Sparsity (%)	Test accuracy (%)	Test loss
1.0	Fully connected	0	87.27	0.384
1.25	4150	35.51	87.60	0.387
1.5	3050	55.12	87.12	0.392
2.0	2420	74.64	86.93	0.399
4.0	1950	93.59	86.68	0.408
8.0	1800	98.37	84.86	0.448

experiment shows that some sparse models can achieve better accuracy than the fully connected model, but accuracy starts degrading when the networks get too sparse.

5.5 Initialization of Structurally Sparse Networks

Verifying the analysis from Sects. 3.1 and 3.2, we compare different initialization strategies and show the results in Table 2.

For the fully connected networks, setting all weights to the same value prevents learning as expected. Similarly, using the same magnitude but setting the sign positive for weights with even index and negative otherwise ("alternating") makes the network too regular to learn. However, we see that simply choosing a random sign can result in very high accuracy, provided the constant w_{init} is chosen carefully as described in Sect. 3.1. We also find that initializing the weights sparse doesn't hurt accuracy, and in fact seems to slightly improve it. With 90% sparsity, it may happen that a weight slice (see Sect. 2.2) is completely zero at initialization. This is not a problem as long as there are non-zero values in the other filters belonging to that neuron.

For the sparse networks, we can make use of the paths for initialization, setting weights belonging to a path with an even index positive and negative otherwise. This can be done only for initialization or permanently in order to save one bit of storage per weight. The last 2 rows of Table 2 show what happens if the signs of the weights are fixed and we only train the weight magnitudes. The signs can be stored or generated dynamically as described in Sect. 3.2. Training only weight magnitudes, while initializing all weights with the same constant, the networks still reach accuracy within 3% of the fully randomly initialized network.

Care needs to be taken when choosing to fix the sign for all weights along a path in a CNN. As a path touches not a single weight but a $w \times h$ depth slice (see Sect. 2.2), enforcing the same sign for all these weights prevents the network from learning many types of features like edges. Still, such a sparse network using random paths

Table 2 Comparison of initializing weights uniformly random, constant positive, constant with half the initial values negative, or constant but positive for odd neuron indices and negative otherwise ("alternating"). The task is image classification (CIFAR-10). The sparse convolutional neural network (CNN) is created by tracing 1024 random paths. These sparse neural networks have 26.4K weights as compared to 70.6K for the dense net. "Signs non-trainable" means that signs are kept fixed after initialization, while training only the weight magnitudes

CNN	Initialization method	Test accuracy (%)
Dense	Uniformly random	87.27
	Constant, positive	10.00
	Constant, alternating sign	10.00
	Constant, random sign	86.88
	Constant, random sign, 90% sparse	87.39
Sparse	Uniformly random	83.71
	Constant, positive	82.15
	Constant, alternating sign	83.70
	Constant, random sign	83.40
	Constant, sign along path	83.75
Sparse, signs fixed (train only magnitude)	Constant, alternating sign	80.77
	Constant with constant sign along path	77.61

is able to reach 80% accuracy. Note that this is not an issue for the common case of 1×1-convolutions, where $w = h = 1$.

The sparse networks are far more robust to the initialization and do not fail in any of the cases even with all weights in a layer set to the same positive constant. Using a deterministic low discrepancy sequence to enumerate the paths and to determine the signs of the weights allows for deterministic initialization and hence brings us one step closer to completely deterministic training.

6 Conclusion

Encoding the network topology by a deterministic low discrepancy sequence brings together quasi-Monte Carlo methods and artificial neural networks. The resulting artificial neural networks may be trained much more efficiently, because they are structurally sparse from scratch. In addition they allow for deterministic initialization. As shown for the example of the Sobol' sequence, the resulting memory access and connection patterns are especially amenable to a hardware implementation, because they guarantee collision-free routing and constant valences across the neural units.

In future work, we will extend the investigations of quasi-Monte Carlo methods applied to other types of neural networks. Especially in the domain of speech recog-

nition, preliminary experiments are very promising. Furthermore, we like to look at more low-discrepancy sequences and at growing neural networks during training by progressively sampling more paths as generated by the low discrepancy sequence.

Acknowledgements The first author is very thankful to Cédric Villani for a discussion on structure to be discovered in neural networks during the AI for Good Global Summit 2019 in Geneva. The authors like to thank Jeff Pool, Nikolaus Binder, and David Luebke for profound discussions and Noah Gamboa, who helped with early experiments on sparse artificial neural networks. This work has been partially funded by the Federal Ministry of Education and Research (BMBF, Germany) in the project Open Testbed Berlin - 5G and Beyond - OTB-5G+ (Förderkennzeichen 16KIS0980).

References

1. Changpinyo, S., Sandler, M., Zhmoginov, A.: The power of sparsity in convolutional neural networks (2017). arXiv:1702.06257
2. Child, R., Gray, S., Radford, A., Sutskever, I.: Generating long sequences with sparse transformers (2019). arXiv:1904.10509
3. Dettmers, T., Zettlemoyer, L.: Sparse networks from scratch: faster training without losing performance. CoRR (2019). arxiv:abs/1907.04840
4. Dey, S., Beerei, P., Chugg, K.: Interleaver design for deep neural networks. In: 51st Asilomar Conference on Signals, Systems, and Computers, pp. 1979–1983. IEEE (2017)
5. Dey, S., Huang, K.-W., Beerel, P., Chugg, K.: Characterizing sparse connectivity patterns in neural networks. In: 2018 Information Theory and Applications Workshop (ITA), pp. 1–9. IEEE (2018)
6. Dey, S., Huang, K.-W., Beerel, P., Chugg, K.: Pre-defined sparse neural networks with hardware acceleration. CoRR (2018). arxiv:abs/1812.01164
7. Dey, S., Shao, Y., Chugg, K., Beerel, P.: Accelerating training of deep neural networks via sparse edge processing. In: Lintas, A., Rovetta, S., Verschure, P.F., Villa, A.E. (eds.) Artificial Neural Networks and Machine Learning—ICANN 2017, pp. 273–280. Springer International Publishing, Cham (2017)
8. Dick, J., Pillichshammer, F.: Digital Nets and Sequences. Cambridge University Press, Discrepancy Theory and Quasi-Monte Carlo Integration (2010)
9. Farhat, N.H., Psaltis, D., Prata, A., Paek, E.: Optical implementation of the Hopfield model. Appl. Opt. **24**(10), 1469–1475 (1985)
10. Frankle, J., Carbin, M.: The lottery ticket hypothesis: finding sparse, trainable neural networks. In: International Conference on Learning Representations (ICLR) (2019)
11. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the fourteenth International Conference on Artificial Intelligence and Statistics, pp. 315–323. JMLR Workshop and Conference Proceedings (2011)
12. Gray, S., Radford, A., Kingma, D.P.: GPU kernels for block-sparse weights (2017). arXiv:1711.09224
13. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
14. Heitz, E., Belcour, L., Ostromoukhov, V., Coeurjolly, D., Iehl, J.C.: A low-discrepancy sampler that distributes Monte Carlo errors as a blue noise in screen space. In: SIGGRAPH'19 Talks. ACM, Los Angeles, United States (2019). https://hal.archives-ouvertes.fr/hal-02150657
15. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)

16. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015)
17. Jayakumar, S., Pascanu, R., Rae, J., Osindero, S., Elsen, E.: Top-KAST: Top-k always sparse training. Adv. Neural Inf. Process. Syst. **33** (2020)
18. Joe, S., Kuo, F.: Remark on algorithm 659: Implementing Sobol's quasirandom sequence generator. ACM Trans. Math. Softw. **29**(1), 49–57 (2003)
19. Joe, S., Kuo, F.: Notes on generating Sobol' sequences. Technical report, School of Mathematics and Statistics, University of New South Wales (2008). http://web.maths.unsw.edu.au/~fkuo/sobol/joe-kuo-notes.pdf
20. Keller, A.: Myths of computer graphics. In: Niederreiter, H. (ed.) Monte Carlo and Quasi-Monte Carlo Methods 2004, pp. 217–243. Springer, Berlin (2006)
21. Keller, A.: Quasi-Monte Carlo image synthesis in a nutshell. In: Dick, J., Kuo, F., Peters, G., Sloan, I. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2012, pp. 203–238. Springer, Berlin (2013)
22. Keller, A., Grünschloß, L.: Parallel quasi-Monte Carlo integration by partitioning low discrepancy sequences. In: Plaskota, L., Woźniakowski, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2010, pp. 487–498. Springer, Berlin (2012). http://gruenschloss.org/parqmc/parqmc.pdf
23. Kriman, S., Beliaev, S., Ginsburg, B., Huang, J., Kuchaiev, O., Lavrukhin, V., Leary, R., Li, J., Zhang, Y.: Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6124–6128 (2020). https://arxiv.org/abs/1910.10261
24. Kundu, S., Nazemi, M., Pedram, M., Chugg, K., Beerel, P.: Pre-defined sparsity for low-complexity convolutional neural networks (2020)
25. Kundu, S., Prakash, S., Akrami, H., Beerel, P., Chugg, K.: pSConv: A pre-defined sparse kernel based convolution for deep CNNs. In: 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 100–107 (2019). https://doi.org/10.1109/ALLERTON.2019.8919683
26. Kung, H., Leiserson, C.: Systolic arrays (for VLSI). SIAM Sparse Matrix Proc. **1978**, 256–282 (1979)
27. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
28. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J.: Importance estimation for neural network pruning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11264–11272 (2019)
29. Mordido, G., Van keirsbilck, M., Keller, A.: Instant quantization of neural networks using Monte Carlo methods. In: NeurIPS 2019 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (NeurIPS 2019 EMC^2) (2019)
30. Mordido, G., Van keirsbilck, M., Keller, A.: Monte Carlo gradient quantization. In: CVPR 2020 Joint Workshop on Efficient Deep Learning in Computer Vision (CVPR 2020 EDLCV) (2020)
31. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. SIAM, Philadelphia (1992)
32. Owen, A.: Randomly permuted (t, m, s)-nets and (t, s)-sequences. In: Niederreiter, H., Shiue, P. (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Lecture Notes in Statistics, vol. 106, pp. 299–315. Springer, Berlin (1995)
33. Paulin, L., Coeurjolly, D., Iehl, J.C., Bonneel, N., Keller, A., Ostromoukhov, V.: Cascaded Sobol' sampling. ACM Trans. Graph. **40**(6), 274:1–274:13 (2021). https://hal.archives-ouvertes.fr/hal-03358957
34. Rui, X., Daquan, H., Zhineng, L.: A perfect shuffle type of interpattern association optical neural network model. Guangzi Xuebao/Acta Photonica Sinica **29**(1) (2000)
35. Rumelhart, D., Hinton, G., Williams, R.: Learning representations by back-propagating errors. In: Anderson, J., Rosenfeld, E. (eds.) Neurocomputing: Foundations of Research, pp. 696–699. MIT Press, Cambridge, MA, USA (1988)

36. Sobol', I.: On the Distribution of points in a cube and the approximate evaluation of integrals. Zh. vychisl. Mat. mat. Fiz. **7**(4), 784–802 (1967). USSR Comput. Math. Math. Phys. 86–112
37. de Sousa, C.: An overview on weight initialization methods for feedforward neural networks. In: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 52–59. IEEE (2016)
38. Stone, H.: Parallel processing with the perfect shuffle. IEEE Trans. Comput. **20**(2), 153–161 (1971)
39. Wächter, C.: Quasi-Monte Carlo Light Transport Simulation by Efficient Ray Tracing. Ph.D. thesis, Universität Ulm (2008)
40. Zhou, H., Lan, J., Liu, R., Yosinski, J.: Deconstructing lottery tickets: Zeros, signs, and the supermask. In: NeurIPS 2019 (2019). Arxiv:1905.01067
41. Zhu, C., Han, S., Mao, H., Dally, W.: Trained ternary quantization. CoRR (2016). arxiv:abs/1612.01064