Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Noah Kriesch

born in Wernigerode (Germany)

2023

Challenging VEGAS with DIRT

This Bachelor Thesis has been carried out by Noah Kriesch at the
Institute for Theoretical Physics in Heidelberg
under the supervision of
Prof. Dr. Tilman Plehn

Abstract

Numerical methods have undergone a significant surge in popularity since the mid-20th century. For particle physics simulations, numerics has become indispensable. Monte Carlo integration, for instance, is employed to solve scattering cross sections. A frequently used integration algorithm is VEGAS, which, despite its applicability, faces some inherent issues. This inspires the search for more efficient sample generation techniques. A recently developed concept, known as Deep Squared Inverse Rosenblatt Transport (DIRT), aims to approximate target functions through functional tensor trains. This thesis examines the feasibility of DIRT as a sampling algorithm for numerical integration in particle physics. As test functions, four-dimensional Gaussian mixtures are employed. The results indicate that DIRT and VEGAS display similar levels of performance. It is worth noting that in some cases a 100 times smaller sample size is sufficient to allow DIRT to achieve comparable accuracy to VEGAS.

Zusammenfassung

Numerische Methoden haben seit Mitte des 20. Jahrhunderts einen enormen Aufschwung erlebt. In der Teilchenphysik sind sie unverzichtbar geworden, um Simulationen durchzuführen. Zum Beispiel wird Monte-Carlo-Integration zur Berechnung von Streuquerschnitten eingesetzt. Ein dabei häufig verwendeter Algorithmus ist VEGAS, der trotz seiner guten Anwendbarkeit mit einigen Problemen zu kämpfen hat. Dies gibt Anlass zur Suche nach effizienteren Verfahren zur Erzeugung von Stichproben. Eine kürzlich entwickelte Idee ist der so genannte Deep Squared Inverse Rosenblatt Transport (DIRT), welcher Funktionen durch Tensor Trains approximiert. In dieser Arbeit wird untersucht, ob DIRT für die numerische Integration in der Teilchenphysik eingesetzt werden kann. Als Testfunktionen dienen vierdimensionale Gaussian Mixtures. Es wird festgestellt, dass DIRT und VEGAS ähnliche Leistungen zeigen. Allerdings genügt DIRT in manchen Fällen eine bis zu 100-mal geringere Stichprobengröße, um gleiche Genauigkeiten wie VEGAS zu erzielen.

Acknowledgements

Writing a thesis proved to be quite an adventure, marked by surprising twists and turns over the course of four months. I am especially grateful to Prof. Tilman Plehn for his invaluable guidance throughout this project. Special thanks go also to Dr. Karina Koval for helping me understand the mathematics behind DIRT and to Prof. Robert Scheichl for inspiring Tilman to supervise a Bachelor thesis on this interesting topic. I would also like to thank Prof. Björn Malte Schäfer willing to provide time and effort as the second examiner of this thesis.

Dear Jonas and Theo, thank you for providing such interesting physical use cases for DIRT and your assistance during implementation. It is a pity that these examples did not allow for more investigation.

To Clara, Marius and Mathis, thank you for your remarkable proofreading efforts, ensuring the clarity and coherence of this work.

I am grateful for the collective support of Tilman's work group. Thank you all for the inspiring lunches and interesting talks!

Contents

1 Introduction

Determining cross-sections of particle interactions in Quantum Field Theory is one of plenty cases where numerical integration can be applied. Improving the computational efficiency can provide more accurate estimates. With this objective it is natural to look for new integration schemes which might enhance the status quo. A frequently applied algorithm is VEGAS[1], originally proposed in 1978 and enhanced in 2021 by Gerard Peter Lepage [9, 10]. It iteratively exploits both importance sampling and since its major update adaptive stratified sampling to perform multidimensional integration. With the 2021 update VEGAS faced large improvements for complex integrand structures, such as non-axis aligned peaks [10]. However, the enhancement was not able to eliminate the causes of these previous problems, as some of them are design-related. This inspires to ask whether mathematical findings of the past decades could introduce even more efficient schemes for particular use cases. In my thesis I take a closer look at one recent idea, namely Deep Squared Inverse Rosenblatt Transport (DIRT). This algorithm was proposed by Tiangang Cui and Sergey Dolgov in 2021 and uses tensor train approximations to iteratively approach the integrand [3]. On the following pages it is investigated whether playing with DIRT can be an actual improvement over VEGAS. For this I compare VEGAS and DIRT on the basis of four-dimensional Gaussian mixtures. Later on, DIRT is challenged by an example from particle physics.

[1]When I name VEGAS in this thesis, I both refer to the classic VEGAS and VEGAS+.

2 Theoretical tools

In this chapter I briefly discuss the conceptual frameworks of both the VEGAS algorithm and the Deep Squared Inverse Rosenblatt Transport (DIRT). While the principal ideas of both algorithms are connected, a closer look reveals that DIRT is based on a much more complex and sophisticated approach than VEGAS. Whether this is of advantage is investigated in the subsequent chapters. I will also provide a rough introduction to the physical example that will be used for practical comparison between VEGAS and DIRT.

2.1 Monte Carlo Methods

For a given non-negative function $f : \Omega \to \mathbb{R}$, the expectation value with respect to an underlying probability density function (p.d.f.) $\pi(x)$ on the domain $\mathcal{D} \subseteq \Omega$ is defined as

$$\mathbb{E}_\pi[f] = \int_\mathcal{D} \mathrm{d}x \, \pi(x) f(x) \, . \tag{2.1}$$

If f itself is a p.d.f. and one wants to determine the probability under f of a random variable X to be in $\mathcal{D}$, the prior π can be constructed such that it is uniform on $\mathcal{D}$ and $\pi(x) = 0$ otherwise.

For larger dimensions integrating over an arbitrary p.d.f. becomes more difficult and analytical solution typically cease to exist. In such cases numerical integration using Monte Carlo methods becomes the mathematical tool of choice. The probably best-known Monte Carlo integration method is the »hit-or-miss« Monte Carlo scheme, where N i.i.d. random points $X_i = (\vec{x}_i, y_i)$ are drawn in the volume $\mathcal{V} \equiv \mathcal{D} \times [\min f(\mathcal{D}), \max f(\mathcal{D})] \subseteq \Omega \times \mathbb{R}$. Then, an integral value estimate can be obtained by determining the fraction of points located underneath the curve created by f and the volume of $\mathcal{V}$,

$$\hat{I}_{MC} = \frac{\#\{X_i = (\vec{x}_i, y_i) \,|\, y_i \leq f(\vec{x}_i)\}}{N} \cdot \mathrm{vol}(\mathcal{V}) \, . \tag{2.2}$$

The second moment

$$\hat{I}^{(2)}_{MC} = \frac{\#\{X_i = (\vec{x}_i, y_i) \,|\, y_i \leq f(\vec{x}_i)^2\}}{N} \cdot \mathrm{vol}(\mathcal{V}) \tag{2.3}$$

then allows to compute the variance of the estimate,

$$\mathrm{var}(\hat{I}_{MC}) = \frac{\hat{I}^{(2)}_{MC} - \hat{I}^{2}_{MC}}{N-1} \,. \tag{2.4}$$

Another slightly more sophisticated approach is to sample N uniformly distributed random points $\vec{x} \in \mathcal{D}$ and then compute the mean of images under f,

$$\hat{I}^{(m)} = \frac{\mathrm{vol}(\mathcal{D})}{N} \sum_{i=1}^{N} f(\vec{x}_i)^m \,, \text{ with variance} \tag{2.5}$$

$$\mathrm{var}(\hat{I}) = \frac{\hat{I}^{(2)} - \left(\hat{I}^{(1)}\right)^2}{N-1} \,. \tag{2.6}$$

Looking at the variances it becomes evident that the accuracy of both approaches converges with $1/\sqrt{N}$ as $N \to \infty$. The computational cost however scales with N. Hence, it is desirable to improve the sampling efficiency before increasing the (absolute) sample size. The most used approach to this is increasing the effective sample size. In the following I will present two Monte Carlo methods which can be applied in such cases and are of relevance to this thesis.

2.1.1 Importance Sampling

In Importance Sampling, a new reference density $g(x)$ (non-negative and non-zero on $\mathcal{D}$) is introduced so that we can write

$$I = \int_{\mathcal{D}} \mathrm{d}x\, f(x) = \int_{\mathcal{D}} \mathrm{d}x\, g(x) \left(\frac{f(x)}{g(x)}\right) \,. \tag{2.7}$$

The samples X_i are now drawn according to g. This requires the introduction of a weight $\omega(x) \equiv \frac{1}{g(x)}$ in the mean of images, yielding a modified formula for the integral estimate

$$\hat{I}^{(m)}_{IS} = \frac{\mathrm{vol}(\mathcal{D})}{N} \sum_{i=1}^{N} \omega(x_i) f(\vec{x}_i)^m \,, \text{ with} \tag{2.8}$$

$$\mathrm{var}(\hat{I}_{IS}) = \frac{\hat{I}^{(2)}_{IS} - \left(\hat{I}^{(1)}_{IS}\right)^2}{N-1}. \tag{2.9}$$

Unsurprisingly the variance is reduced if g is similar to f, i.e. if the Hellinger error

$$D_H[f,g] = \frac{1}{\sqrt{2}} \int_{\mathcal{D}} \mathrm{d}x \left(\sqrt{f(x)} - \sqrt{g(x)}\right)^{1/2} \tag{2.10}$$

is reduced. The efficiency of variance reduction can be measured by calculating the Effective Sample Size ESS, defined as

$$\mathrm{ESS}[\omega; N] \equiv \frac{N}{1 + \mathrm{var}(\omega(x))} \,. \tag{2.11}$$

2.1.2 Stratified Sampling

Assuming that only a small fraction of the integrated domain is of interest for a good estimate of the integral (which typically holds for large dimensions), it makes sense to concentrate the drawn samples on these regions of interest. Stratified Sampling can be applied to achieve this. Initially one creates a disjoint partition of $\mathcal{D}, \mathcal{D} = \bigcup_{j=1}^{K} \mathcal{D}_j$. Then, $N_j = \lfloor \frac{N}{K} \rfloor$ samples are drawn for each partition $\mathcal{D}_j$ and the estimate and variance of the local integral

$$I_j = \int_{\mathcal{D}_j} \mathrm{d}x\, f(x) \tag{2.12}$$

is determined. In the following iterations the number of samples N_j is varied so that the sum of relative variances $\sum_{j=1}^{K} \frac{\hat{I}_j}{\sigma_{\hat{I}_j}}$, and thus the overall variance $\mathrm{var}(\hat{I})$ is minimized. The optimum stratification has been achieved, $\frac{\hat{I}_j}{\sigma_{\hat{I}_j}} = \frac{\hat{I}_i}{\sigma_{\hat{I}_i}}\ \forall i, j = 1, \ldots, K$.

2.2 VEGAS

VEGAS was first invented by G. P. Lepage in 1978. He aimed to provide a general sampling algorithm for efficient integration in initially four or more dimensions [9]. The original, »classic« VEGAS applies adaptive importance sampling to better approximate functions in numerical integration. This approach lead to some issues by design - it faced huge problems to resolve structures which were non-trivial in terms of axis-alignment, e.g. a Gaussian mixture of peaks placed along the diagonal of a hypercube [10]. To understand where this problem originated, I will first discuss the classic VEGAS algorithm and take a brief look onto the causes. Based on this introduction I then continue to describe the enhanced »VEGAS+« algorithm, which has been introduced by Lepage in 2021 [10].

2.2.1 Classic VEGAS

The underlying idea of VEGAS is to design an integration algorithm using adaptive importance sampling to approximate the target distribution. In the following I describe this approach for the one-dimensional case. The example is identical to the one given in [9] and can be applied for multiple dimensions in an analogous manner.

Suppose one aims to determine the estimate $\hat{I}$ of an integral

$$I[f] = \int_a^b \mathrm{d}x\, f(x)\ ,$$

with f being a non-negative function on $[a, b] \subset \mathbb{R}$. A useful approximate density in this case is a step function on $[a, b]$. A set of equally spaced grid points $\{x_i \,|\, i = 1, .., M\} \in [a, b]$ is constructed, demanding $x_1 = a \wedge x_M = b$. To simplify the notation, the increments $\Delta x_i \equiv x_i - x_{i-1}$ are introduced along with

$$p^{(s)}(x) = \frac{1}{M} \cdot \frac{1}{\sum_{i=1}^{M-1} \Delta x_i^{(s)} \Theta(x - x_i^{(s)}) \Theta(-x + x_{i+1}^{(s)})} \tag{2.13}$$

as the approximate importance density (step) function for step s. Then, an iterative procedure is followed.

In each step s the grid is adjusted as follows. N independent random samples $y_j \in [a, b]$, $y_j \sim p^{(s-1)}(x)$ are drawn, leaving $\sim \lfloor N/M \rfloor$ samples per increment. Every increment is again divided into $m_i + 1$ sub-increments, where

$$m_i \equiv \text{const.} \cdot \left(\frac{\bar{f}_i \Delta x_i}{\sum_k \bar{f}_k \Delta x_k} \right) , \tag{2.14}$$

and $\bar{f}_i \equiv \sum_{y_j \in (x_{i-1}, x_i)} |f(y_j)|$. Then, new increments Δx_i are constructed so that all increments contain an equal number of sub-increments. This way, after several iterations Δx_i will increase in size where $f(x)$ is small and shrink where $f(x)$ exhibits larger values. Hence, $p^{(s)}$ will come to better approximate f. To complete each step, for the N drawn random samples $y_j \sim p^{(s)}(x)$, the new integral estimate is computed as

$$\bar{I}_s^{(m)} = \frac{(b-a)}{N} \sum_{j=1}^{N} \frac{f(y_j)^m}{p^{(s)}(y_j)} \text{ , with} \tag{2.15}$$

$$\sigma_s^2 \simeq \frac{\bar{I}_s^{(2)} - \left(\bar{I}_s^{(1)}\right)^2}{N-1} \text{ .} \tag{2.16}$$

For an increasing sample size N the estimates $\bar{I}_s$ follow a Gaussian distribution and can then be used to give a cumulative weighted estimate $\hat{I}$ [10],

$$\hat{I} = \frac{\sum_s \frac{\bar{I}_s}{\sigma_s^2}}{\sum_r \frac{1}{\sigma_r^2}} \text{ with} \tag{2.17}$$

$$\sigma_{\hat{I}} = \hat{I} \left(\sum_s \frac{1}{\sigma_s^2} \right)^{-1/2} \tag{2.18}$$

Without loss of generality this concept can be applied to the multidimensional case in a similar manner [9].

2.2.2 The Problem with VEGAS

The grid constructed by VEGAS becomes impractical for medium-complex structures in multiple dimensions, e.g. a Gaussian mixture which is not aligned along one of the axes. A very simple but illustrative example can be given by the following toy function in (only!) 2 dimensions:

$$g(\vec{x};\vec{\mu}_i,\sigma^2) = \frac{1}{\sqrt{8\pi \det \mathbf{C}}} \sum_{i=1,2} \exp\left[(\vec{x}-\vec{\mu}_i)^\top \mathbf{C}^{-1} (\vec{x}-\vec{\mu}_i)\right] \ , \text{ where}$$
$$\vec{\mu}_i = \begin{pmatrix} i/3 \\ i/3 \end{pmatrix} \quad , \quad \mathbf{C} = \operatorname{diag}\left(\sigma^2,\sigma^2\right) \ . \tag{2.19}$$

When integrating over the domain $[0,1]^2$, VEGAS draws large numbers of samples in the regions around $(1/3,\ 1/3)^\top$ and $(2/3,\ 2/3)^\top$, but will make the same effort for the phantom peaks at $(1/3,\ 2/3)^\top$ and $(2/3,\ 1/3)^\top$ (c.f. figure 2.1). For d dimensions the computational complexity grows with 2^d for only two regions of interest, leaving us with $2^d - 2$ surplus regions of high sampling density. In other words this approach would yield an effective sample size of only $N/2^{d-1}$. This becomes clearly not sensible for a medium or large number of dimensions.

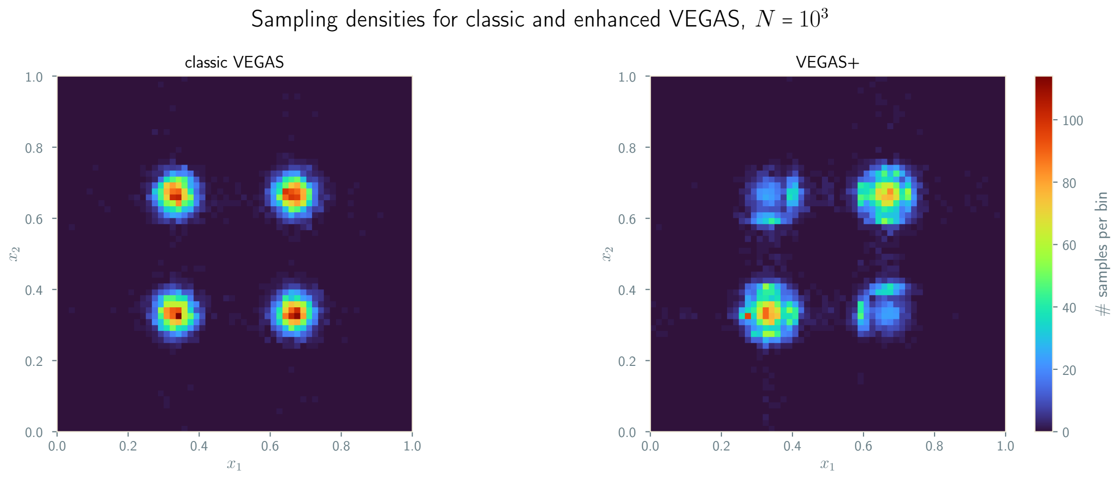

Figure 2.1: Sampling densities from classic VEGAS and VEGAS+ for the Gaussian mixture model from eqn. (2.19) with a variance of $\sigma^2 = 10^{-3}$. One can see the phantom peaks in the left plot and their remnants to the right. Both algorithms give similar results, $\hat{I}_{VEGAS} = 0.9940(27)$ and $\hat{I}_{VEGAS+} = 1.000\,05(29)$.

2.2.3 VEGAS+

To circumvent this weakness, Lepage proposed in 2021 to enhance VEGAS with an adaptive stratified sampling algorithm [10]. To achieve this, each axis is partitioned into a number of N_{st} stratifications, yielding N_{st}^d hypercubes in the integration domain. Prior to the grid adjusting step the number of required samples (as a fraction of the total number of N samples) is determined for each hypercube. This allows to exclude regions of low interest which would have been included before by pure adaptive importance sampling.

However, for this to have any effect on the results a minimum of $N_{st} = 2$ stratifications per axis is required. This yields a constraint for the minimum number of samples, namely $N \overset{!}{\geq} 4N_{st}^d \geq 2^{d+2}$, where the factor of 4 is more or less arbitrary but needed to ensure that the number of samples per hypercube can be varied [10]. For small-dimensional use-cases, VEGAS+ thereby offers a practicable solution. However, as Lepage also points out himself, for medium-high and large dimensions the considered approach runs into a dead end. He provides the example that for 25 dimensions the sample size needs to be larger than 1.3×10^8 to allow adaptive stratified sampling.

Lepage implements mixed stratified sampling as a viable solution to this issue, i.e. the option to vary the number of stratifications for different axes. While this approach can suffer from phantom peaks, it seems to perform nearly as good as the unmixed stratified sampling. Overall, it appears that VEGAS+ outperforms classic VEGAS by results which are 2 - 18 times more accurate [10].

2.3 (D)IRT

A different approach to a generalised sampling algorithm is taken by Cui et al. [3, 5]. They propose a combination of generalised inverse Rosenblatt transports and tensor-train approximations to construct the importance density iteratively. The final product of their considerations is called Deep Squared Inverse Rosenblatt Transport (DIRT) and in contrast to VEGA allows natively to adapt easily to, e.g., non-axis-aligned structures. To understand this algorithm, I first will briefly discuss the original Rosenblatt transport, its inverse and tensor-trains based construction of such transport maps. Then, I will cover how the Squared Inverse Rosenblatt Transport (SIRT) and DIRT work.

2.3.1 Rosenblatt Transport and Functional Tensor Trains

The Rosenblatt transport constructs a diffeomorphism T that allows to map an arbitrary but absolutely continuous d-variate distribution into the uniform distribution on the d-dimensional unit hypercube $\mathcal{U} = [0,1]^d$ [15]. In this section, I follow the descriptions by Rosenblatt [15] and Cui et al. [3] to explain the mathematical background.

Let $X = (X_1, \dots, X_d) \in \mathcal{X} \equiv \mathcal{X}_1 \times \dots \times \mathcal{X}_d \subseteq \mathbb{R}^d$ be a random vector with the p.d.f. $f_X(x_1, \dots, x_d)$ and the respective cumulative distribution function $F(x_1, \dots x_d) = \int_{-\infty}^{x_1,\dots,x_d} \mathrm{d}x_1' \dots \mathrm{d}x_d' \, f_X(x_1', \dots, x_d')$. Then, the Rosenblatt transport $T : \mathcal{X} \to \mathcal{U}$, $x \mapsto z = Tx$ is given by a series of marginal cumulative distribution functions

$$\begin{cases} z_1 & = \mathbb{P}(X_1 \le x_1) = F_{X_1}(x_1) \\ z_2 & = \mathbb{P}(X_2 \le x_2 | X_1 = x_1) = F_{X_2|X_1}(x_2|x_1) \\ & \vdots \\ z_d & = \mathbb{P}(X_d \le x_d | X_i = x_i \, \forall \, i = 1, \dots, d) = F_{X_d|X_{<d}}(x_d|x_{d-1}, \dots, x_1) \, . \end{cases} \tag{2.20}$$

One can then prove, by simple integration, that the random variable $Z = TX$ is uniformly distributed on $\mathcal{U}$.

As mentioned earlier the Rosenblatt transport is a diffeomorphism. Consequently, T can be inverted, allowing to use samples $Z \in \mathcal{U}$ to calculate their corresponding random values $X \in \mathcal{X}$, $X = T^{-1}Z$. This turns out to be quite useful, as it is much easier to sample from a uniform than an arbitrary distribution. The inverse of T is described as

$$x = T^{-1}z = \left[F_{X_1}^{-1}(z_1),\, F_{X_2|X_1}^{-1}(z_2|x_1), \dots,\, F_{X_d|X_d}^{-1}(z_d|x{<}\, d)\right]^{\top} \tag{2.21}$$

and is hence called Inverse Rosenblatt Transport (IRT).

The second but more important underlying concept for this thesis is that of tensor trains (TT), presented in [12] by Oseledets. Let $\mathfrak{T}$ be a d-dimensional $n_1 \times n_2 \times \cdots \times n_d$ tensor. $\mathfrak{T}$ has TT-format if its elements can be written as

$$\mathfrak{T}_{i_1,\dots,i_d} = \sum_{\alpha_0,\dots,\alpha_d} \mathbf{G}^{(1)}_{\alpha_0,i_1,\alpha_1} \mathbf{G}^{(2)}_{\alpha_1,i_2,\alpha_2} \cdots \mathbf{G}^{(d)}_{\alpha_{d-1},i_d,\alpha_d} \tag{2.22}$$

and the so-called cores $\mathbf{G}^{(k)}$ are of size $r_k{-}1 \times n_k \times r_k$, with $k = 1, \dots, d$, $r_0 = r_d = 1$. In a shorthand notation equation (2.22) can also be written as

$$\mathfrak{T}_{i_1,\dots,i_d} = \mathbf{G}^{(1)}_{i_1} \mathbf{G}^{(2)}_{i_2} \cdots \mathbf{G}^{(d)}_{i_d}\,. \tag{2.23}$$

In [12] Oseledets proves that any d-dimensional tensor can be approximated by a TT-format tensor up to arbitrary accuracy. This algorithm is called TT singular value decomposition and provides a practical analytical approach to approximate tensors for computational operations [12].

To further understand how this transformation can be combined with so-called functional tensor trains, it is practical to adapt to the notation of Cui et al. in [3]. They write the target p.d.f. in the form

$$f_X(x) = \frac{1}{\mathcal{N}} \cdot \underbrace{\pi(x)}_{\text{unnormalized density}} \cdot \overbrace{\lambda(x)}^{\text{weighting function}} \quad , \quad \mathcal{N} = \int_{\mathcal{X}} \mathrm{d}x\, \pi(x)\lambda(x)\,, \tag{2.24}$$

given that $\pi \in L^1_\lambda(\mathcal{X})$, $\pi(x) \geq 0\ \forall\, x \in \mathcal{X}$. Arguing that multivariate functions are a continuous analogue of tensors, it is possible construct the TT approximation for a general multivariate function $\pi : \mathcal{X} \to \mathbb{R}$, $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$. In matrix notation, $\pi(x)$ can thus be written as the product of TT-cores $\mathbf{H}_k(x_k) : \mathcal{X} \to \mathbb{R}^{r_{k-1} \times r_k}$,

$$\pi(x) \simeq \tilde{\pi}(x) \equiv \prod_{k=1}^{d} \mathbf{H}_k(x_k) \;\text{ with} \tag{2.25}$$

$$\mathbf{H}_k(x_k) \equiv \left[\sum_{i=1}^{n_k} \phi_k^{(i)}(x_k) \mathfrak{A}_k^{\alpha,i,\beta}\right]_{\alpha=1,\dots,r_{k-1}\,,\beta=1,\dots,r_k}\,, \tag{2.26}$$

given a set of basis functions $\{\phi_k^{(1)}(x_k), \dots, \phi^{(n_k)}\}$ and a respective coefficient tensor $\mathfrak{A}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$. The summation ranges n_k are called TT ranks. For $k = 1, d$ they are limited to $n_0 = n_d$ (default choice being $n_0 = 1$). Otherwise, they can be chosen freely. The TT-approximation $\tilde{\pi}$ is determined numerically and in theory exact. However, due to limitations in TT ranks and the choice of basis functions $\tilde{\pi}$ usually remains an approximation of π.

Assuming that the approximate TT-decomposition of $\pi(x)$ is given as

$$\tilde{\pi}(x_1, \dots, x_d) = \mathbf{H}_1(x_k) \cdots \mathbf{H}_d x_d \,, \tag{2.27}$$

the corresponding approximate target p.d.f. then is $\tilde{f}_X(x) = \frac{1}{\tilde{c}}\tilde{\pi}(x)\lambda(x)$, with the normalizing constant set to $\tilde{c} = \int_{\mathcal{X}} \mathrm{d}x\, \tilde{\pi}(x)\lambda(x)$. Defining

$$\bar{\mathbf{H}}_k \equiv \int_{\mathcal{X}_k} \mathrm{d}x_k\, \mathbf{H}_k(x_k)\lambda_k(x_k) \,, \tag{2.28}$$

$$\tilde{\pi}_{\leq k}(x_{\leq k}) = \mathbf{H}_1(x_1) \cdots \mathbf{H}_k(x_k)\bar{\mathbf{H}}_{k+1} \cdots , \bar{\mathbf{H}}_d \tag{2.29}$$

the k-th marginal p.d.f. is found to be

$$\tilde{f}_{X_{\leq k}}(x_{\leq k}) = \frac{1}{\tilde{d}}\tilde{\pi}_{\leq k}(x_{\leq k}) \prod_{i=1}^{k} \lambda_i(x_i) \quad , \tilde{d} = \bar{\mathbf{H}}_1 \cdots \bar{\mathbf{H}}_d \,. \tag{2.30}$$

With that in mind one can write down the conditional probabilistic densities, define a Rosenblatt transport and the corresponding IRT. This way one is able to draw random variables following a uniform distribution and obtain a random variable with p.d.f. $\tilde{f}_X$. Defining n as the maximum number of basis functions, r as the maximum TT rank used and assuming that N samples are drawn, Cui et al. find that the total numerical complexity can be estimated to be $\mathcal{O}(dnr^2 + dNr^2 + Ndnr)$ [3].

2.3.2 SIRT

This functional tensor-train approach to IRT (TT-IRT) can face the problem that the approximated p.d.f. may not be positive on the whole domain due to rank truncations. Cui et al. solve this problem by approximating $\sqrt{\pi(x)}$,

$$\sqrt{\pi(x)} \simeq \tilde{g}(x) = \prod_{i=1}^{d} \mathbf{G}_i(x_i) \,, \tag{2.31}$$

instead of $\pi(x)$ directly. Then, the target p.d.f. is approximated by the estimate

$$\hat{f}_{\hat{X}}(x) = \frac{1}{\hat{c}}\hat{\pi}(x)\lambda(x) \,, \quad \begin{array}{l} \hat{\pi}(x) = \gamma + \tilde{g}(x)^2 \\ \hat{c} = \gamma\lambda(\mathcal{X}) + \int_{\mathcal{X}} \mathrm{d}x\, \tilde{g}(x)^2\lambda(x) \end{array} , \tag{2.32}$$

with $\gamma > 0$ as a correction constant. The so called Squared Inverse Rosenblatt Transport (SIRT) then is obtained similarly to TT-IRT. Compared to TT-IRT, SIRT can ensure to preserve smoothness and monotonicity [3]. However, this comes at the cost of increasing complexity now scaling with increased leading terms, $\mathcal{O}(dnr^3 + Ndnr^2 + Ndr^2)$ [3].

2.3.3 DIRT

The capability of SIRT to adapt to complicated structures mainly depends on the allowed maximum rank for truncations. As could be seen, the IRT-algorithms scale at least quadratically with the ranks. Hence, a direct factorisation can become costly. The Deep Squared Inverse Rosenblatt Transport (DIRT) builds a composition of SIRT mappings to solve this.

Define a sequence of bridging measures π_k, $k = 1, \dots, L$ and set $\pi_L \equiv \pi$. Then, equivalently to equation (2.32), it is

$$f_{X^k}(x) = \frac{1}{c_k}\pi_k(x)\lambda(x) \;, \quad c_k = \int_{\mathcal{X}} \mathrm{d}x\, \pi_k(x)\lambda(x) \;. \tag{2.33}$$

In each step $f_{X^k}(x)$ is approximated by a SIRT T_k. This yields a composition describing the k-th bridging density $(T_0 \circ T_1 \circ \dots \circ T_k)_\sharp g_{\text{uniform}}(x) = \pi_k(x)$. By gradually increasing the complexity of the bridging densities, this algorithm allows to describe more and more complicated structures. A sensible choice for the bridging densities are tempered distributions, i.e. $\pi_k(x) = \pi(x)_k^\beta$, $0 \le \beta_1 < \dots < \beta_L = 1$ [3]. This way, for the initial k the difficult structures in the distribution are smeared out, allowing DIRT to adapt to an approximated version of the integrand first. Then, with growing index k the algorithm will be able to »learn« the actual distribution function[3].

2.4 Physical motivation

The current Standard Model of particle physics works surprisingly well. However, experiments indicate some possible flaws of the Standard Model (SM), such as the $(g-2)_\mu$-anomaly [1], the Hubble tension [18] or Dark Matter [14]. All of these phenomena can be described as consequences of modifications of the SM [11]. One candidate for such a modification is the extension of the SM by a dark fermion singlet χ and a vector mediator Z' coupling SM leptons ℓ and the fermion singlets χ [11]. This modification can be expressed by an additional term in the SM Lagrangian,

$$\mathcal{L}_{SM} \to \mathcal{L}_{SM} + Z'_\mu \left(g_\ell \bar{\ell}\gamma^\mu \ell + g_\chi \bar{\chi}\gamma^\mu \chi \right) \;, \tag{2.34}$$

where g_ℓ, g_χ denote the respective coupling parameters[1]. The challenge is now to gain estimates of these couplings. An interesting approach to this is proposed by Manzari et al. in [11]. From neutrino flux measurements originating from core-collapse supernovae (SN) the total emitted energy can be determined. Given the assumption that the vector mediators can be produced on-shell in stellar plasmas, the modification predicts that $\chi\bar{\chi}$-pair will be created by $\ell\bar{\ell}$-annihilation processes and photoproduction [11]. In this thesis I limit myself to the case of annihilation. The corresponding Feynman diagram is shown in figure 2.2. The known classical luminosities determined by the neutrino flux measurements set upper boundaries for the corresponding dark luminosity. This allows to set constraints for the couplings g_ℓ, g_χ with respect to the masses

[1]The complete extension includes a further term describing the interaction between SM neutrinos and the vector boson Z' [11], which is of no relevance here.

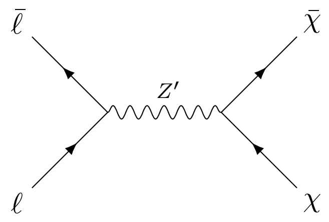

Figure 2.2: Feynman diagram of the first order annihilation process $\ell\bar{\ell} \to \chi\bar{\chi}$.

$m_{Z'}, m_\chi$. This argument allows to cross-check the proposed theory with non-terrestrial data and is also expected to give better constraints than current experiments on earth could [11].

Following Manzari et al. in [11] the differential luminosity can be expressed as

$$\frac{\mathrm{d}Q}{\mathrm{d}R} = \frac{1}{4\pi^4}\int_{m_\ell}^{\infty}\int_{m_\ell}^{\infty}\int_{-1}^{1}\mathrm{d}E_1\,\mathrm{d}E_2\,\mathrm{d}\cos\theta\,\frac{(E_1+E_2)\bar{P}(E_1)\bar{P}(E_2)}{F(E_1;\mu_\ell)F(E_2;-\mu_\ell)}s\sqrt{1-4\frac{m_\ell^2}{s}}\sigma(s)\,, \quad (2.35)$$

with $\bar{p}(E_i)$ as the absolute three-momenta of the the annihilating particles and their respective Fermi-Dirac-distributions $[F(E_1;\mu)]^{-1}$, where μ denotes the chemical potential of the annihilating lepton,

$$\bar{p}(E_i) \equiv \sqrt{E_i^2 - m_\ell^2} \quad \text{and} \quad (2.36)$$

$$F(E_i;\mu) \equiv \exp\left[\frac{E_i-\mu}{T}\right] + 1\,. \quad (2.37)$$

The Mandelstam variable

$$s = 2\left(m_\ell^2 + E_1E_2 - \sqrt{E_1^2-m_\ell^2}\sqrt{E_2^2-m_\ell^2}\cos\theta\right) \quad (2.38)$$

allows a more concise formula for the cross-section,

$$\sigma(s) = \frac{g_\ell^2 g_\mu^2}{12\pi}\frac{s}{(s-m_{Z'}^2)^2+m_{Z'}^2\Gamma_{Z'}^2}\frac{\beta_\chi(s)}{\beta_\ell(s)}\kappa_\ell(s)\kappa_\chi(s)\,,\ \text{given that} \quad (2.39)$$

$$\beta_i(s) \equiv \sqrt{1-4\frac{m_i^2}{s}}\,, \quad (2.40)$$

$$\kappa_i(s) \equiv \left(1+2\frac{m_i^2}{s}\right) \quad \text{and} \quad (2.41)$$

$$\Gamma_{Z'} = \frac{m_{Z'}}{12\pi}\sum_{i=\ell,\chi} g_i^2\kappa_i(m_{Z'}^2)\beta_i(m_{Z'}^2)\Theta(m_{Z'}-2m_i)\,. \quad (2.42)$$

Using the data of the SN 1987A from [2], the computations in this thesis are performed for $T = 36.8\,\mathrm{MeV}$, $\mu_\mu = 102.5\,\mathrm{MeV}$.

After a closer look at the integrand equation (2.35) becomes interesting for DIRT and VEGAS. The propagator in the cross-section term produces a narrow peak at $\sqrt{s} = m_{Z'}$ of width $\Gamma_{Z'} \sim g_i^2 m_{Z'}$. For smaller couplings, VEGAS is unable to resolve this peak without further information, at least for couplings $g \equiv g_\ell = g_\chi \lesssim 10^{-2}$. It is therefore interesting to investigate whether

DIRT might be better able to detect the peak and provide correct estimates. To simplify the computations, the integrand is rendered dimensionless by substituting[2]

$$E_i \to x_i \equiv \frac{E_i}{T} \quad s \to y \equiv \frac{s}{T^2} \quad m_j \to x_j \equiv \frac{m_j}{T} \; . \tag{2.43}$$

This transformation leads to an additional factor of T^7 in front of the integral.

In order to directly resolve the peak in the integration variables, a change of coordinates $\cos\theta$ to y is performed. Without this change VEGAS would be unable to detect the peak [11]. For future research it would be interesting to test whether DIRT is capable of resolving the peak without a change of variable. The substitution gives an additional Jacobian of

$$J_y \equiv \frac{1}{2\bar{p}(x_1)\bar{p}(x_2)} \; , \tag{2.44}$$

where $\bar{p}$ denotes the dimensionless equivalent of $\bar{P}$, i.e. $\bar{p}(x_i) \equiv \sqrt{x_i^2 - x_\ell^2}$. The same goes for f, $f(x_i,\mu) \equiv \exp\left[x_i - {}^{\mu}\!/\!{}_{T}\right] + 1$. This yields a new expression for equation (2.35),

$$I = \frac{T^7}{4\pi} \int_{x_\ell}^{\infty} \int_{x_\ell}^{\infty} \int_{y_a}^{y_b} \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}y\, \frac{(x_1 + x_2)\bar{p}(x_1)\bar{p}(x_2)}{f(x_1;\mu_\ell) f(x_2;-\mu_\ell)} \sqrt{1 - 4\frac{x_\ell^2}{y}}\, \sigma'(y) J_y \; , \tag{2.45}$$

with $\sigma'(s) \equiv s\sigma(s)$. The boundaries y_a, y_b are defined by the physical condition that $y \overset{!}{\geq} 4 \max x_\chi^2, x_\ell^2 \equiv y_{min}$, otherwise the annihilation process would not take place. The formula (2.38) further confines the domain of y, as $\cos\theta \in [-1, 1]$. This results in

$$y_a \equiv \max(y_{min}, y_-) \qquad\qquad y_b \equiv \max(y_{min}, y_+) \; , \text{ where} \tag{2.46}$$

$$y_\pm \equiv 2(x_\ell^2 + x_1 x_2 \pm \bar{p}(x_1)\bar{p}(x_2)) \; . \tag{2.47}$$

As a last step I implement substitutions so that the integration variables can be drawn from finite intervals, i.e. from $[0, 1)$ by defining

$$x_i \equiv (z_i) = \frac{z_i}{1 - z_i} \quad ; \quad y \equiv y(z_y) = \frac{z_i}{1 - z_i} \quad ; z_j \in [0, 1) \; .$$

This change of variables leaves us with an additional Jacobian

$$J_1 = \frac{1}{(1 - z_1)^2} \frac{1}{(1 - z_2)^2} \frac{1}{(1 - z_y)^2} \; . \tag{2.48}$$

[2]These and the following substitutions are identical to the approach taken in [17]. I do not consider all of their choices optimal, but decided to stick as closely as possible to their notation for the sake of clarity.

The resulting integral has the form

$$\frac{\mathrm{d}Q}{\mathrm{d}V} = \frac{T^7}{4\pi}\int_{z_\ell}^{1}\int_{z_\ell}^{1}\int_{z_a}^{z_b}\mathrm{d}z_1\,\mathrm{d}z_2\,\mathrm{d}z_y\,\frac{(x_1+x_2)\bar{p}(x_1)\bar{p}(x_2)}{f(x_1;\mu_\ell)f(x_2;-\mu_\ell)}\sqrt{1-4\frac{x_\ell^2}{y}}\sigma'(y)J_yJ_1 \tag{2.49}$$

$$z_\ell \equiv \frac{x_\ell}{1+x_\ell}\ , \tag{2.50}$$

$$z_{a,b} \equiv \frac{y_{a,b}}{1+y_{a,b}}\ . \tag{2.51}$$

Whether the peak is resolved can be tested by plotting $\mathrm{d}Q/\mathrm{d}V \cdot g^{-4}$. If the peak has been found, it dominates the integral. Then, the cross-section scales $\sigma \propto g^{-2}$. Otherwise, the first term in equation (2.39) dominates, yielding $\sigma \propto g^{-4}$. The change from $\mathrm{d}Q/\mathrm{d}R$ to $\mathrm{d}Q/\mathrm{d}V$ is performed by computing

$$\frac{\mathrm{d}Q}{\mathrm{d}V} = \frac{\mathrm{d}Q}{\mathrm{d}R}4\pi R^2\ . \tag{2.52}$$

For the chosen setting, it is $R \simeq 9.22\,\mathrm{km} \simeq 4.70\times 10^{16}\,\mathrm{MeV}^{-1}$.

3 Implementing DIRT

In the beginning of this thesis the naïve aim was to implement DIRT in Python. This idea arose due to several reasons, the main one being the widespread usage of Python in physics. However, the currently available realisation of DIRT is written in MATLAB [4, 6]. Together with that, only a basic implementation of IRT in Python is presented [6]. A direct implementation of DIRT in Python would allow easier dissemination of the algorithm in Physics community. Respectively, working in Python could have allowed for huge time savings, as the main reason for bugs and error messages originated in the attempt to implement code written in different programming languages into one another. Furthermore, there exists no implementation of VEGAS in MATLAB but in Python. Testing both algorithms in the same framework would have been the most practical approach. However, this goal could not be met.

This chapter explains the final solution and attempts to outline the previous approaches to implementing DIRT in Python (DIRTipy).

3.1 DIRT in MATLAB, VEGAS in Python

The solution found for this project is to run DIRT and VEGAS separately, DIRT in MATLAB and VEGAS in Python. For VEGAS I use the official Python package [10]. The integrands are written using the `numpy` package [7]. Accelerated processing in VEGAS is ensured by adding the `@vegas.batchintegrand` decorator in front of the respective functions.

For DIRT I use the TT-IRT package created by Sergey Dolgov [3, 6], which also requires the TT-Toolbox [13]. The integral estimate is obtained by computing the mean of the transformed samples, returned by `tt_dirt_sample()`. A minimal example of the DIRT code can be found in the appendix (7). Contrary to VEGAS, DIRT comes with a range of tunable parameters. For this thesis I decided to mainly investigate the influence of the sample size N, the grid size n and the initial TT-rank R_0. Unless otherwise mentioned, I choose the tempering powers to be $\log \beta_k \equiv -4 + 1,\ k = 0, \dots, 4$. This is coarser than the choice made by Cui et al. in [3], but should suffice for most of the presented examples. Furthermore, I set the reference density to `reference` = 'n1.5' (truncated Gaussian density with a support on $(-1.5, -1.5)^d$).

My aim is to achieve a broad overview over the modes of operation of DIRT and VEGAS. Thus, DIRT and VEGAS are run only once for each combination of parameters. If the algorithms do not succeed to yield a result, they are run once more before leaving out this data point. Otherwise, recording the data would have required to much time for this Bachelor thesis' project. For future research it is advisable to select single settings, for which DIRT and VEGAS are run multiple times so that statistical statements can also be made, e.g. on the ability of peak identification.

3.2 Attempts to DIRTipy

During this project I made several attempts to create an interface of DIRT that could be run from Python. None of these attempts were really successful nor were the attempts to use Python functions as integrands in MATLAB[1]. This sections aim is to briefly document the attempts I took and thereby hopefully help similar future projects to be of more success.

In theory MATLAB / MathWorks provides the option to build Python packages from MATLAB code. In practice this works excellently for simple applications, such as simple matrix operations, but leads to problems when functions are passed as arguments. A huge problem that I had to face during this thesis project was the frugal documentation of MATLAB packages. The approaches to solutions were unnecessarily hindered by this, eventually leading up to a point where I had to discard the idea of a MATLAB generated Python package for DIRT. Still following the approach to run both VEGAS and DIRT from Python, I found the rather cumbersome solution to call DIRT as a Matlab script through a shell command. This worked but proved to be highly inefficient. Compared to a pure MATLAB solution, the Shell approach exhibited an increased runtime by a factor of up to 75.

3.3 Implementation of the physical case study

As a case study for its implementation in physics I compare DIRT and VEGAS on the problem described in section 2.4. I used the Python code provided in [17, 11] for testing in VEGAS. To write the MATLAB version of the integrand, I used this code as a basis.

[1]However, the latter one might have been caused by a too small knowledge of MATLAB at this stage. I later tried again to implement a Python function in MATLAB with more success considering the runtime efficiency.

4 Discussion of results

To get an impression on how DIRT works, which parameters are the most relevant and which issues might have to be tackled, I test DIRT using Gaussian mixtures (GM) as toy examples. The tests are performed in four dimensions. In the end, I will compare VEGAS and DIRT on the basis of the integral in equation (2.35).

In my comparisons, I focus mainly on the quality of the result, but leave out run times. This is due to the fact that I do not consider the current implementations of DIRT and VEGAS comparable in terms of performance.

4.1 Gaussian Mixtures in four dimensions

Gaussian mixtures are appropriate toy models to imitate arbitrary distributions with sharp peaks. For the classic VEGAS, diagonally aligned Gaussian mixtures proved to be already challenging. This issue has been partially resolved with the new VEGAS+ and should hence not be of relevance in small dimensions. However, the question remains as to how well VEGAS can resolve small peak distributions. I will concentrate on peak widths of $\sigma \sim 0.1 - 0.01$ ($\sigma^2 \sim 10^{-2} - 10^{-4}$) for a integration domain of $[-1,1]^d$. These widths are similar to typical peak bandwidths in Quantum Field Theory and Particle Physics. As shown later on it is no challenge for both DIRT and VEGAS to integrate in the case of $\sigma^2 = 10^{-2}$. Thus, it is more interesting to look at narrower distributions. For this thesis I decided to examine the case of $\sigma^2 = 10^{-3}$ closely. The cases of $\sigma^2 = 10^{-2}, 10^{-4}$ will also be covered but in less detail.

4.1.1 Diagonal Gaussian mixture with σ^2=10^{-3}

Given the case of a simple, normalized diagonal Gaussian mixture,

$$\begin{aligned} g(\vec{x};\vec{\mu}_i,\mathbf{C}) &= \sum_{i=0,1,2} \frac{1}{\sqrt{(2\pi)^d \det \mathbf{C}}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^\top \mathbf{C}^{-1}(\vec{x}-\vec{\mu}_i)\right) \text{ where} \\ \vec{\mu}_i &\equiv \frac{-1+i}{2}\cdot\begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix}^\top \quad , \quad \mathbf{C} \equiv \sigma^2 \cdot \mathbb{I}_4 , \end{aligned} \tag{4.1}$$

I initially choose $\sigma^2 \overset{!}{=} 10^{-3}$ and define the error of an estimate $\hat{I}$ as $\Delta(\hat{I}) \equiv |I_{\text{true}} - \hat{I}| = |1 - \hat{I}|$.

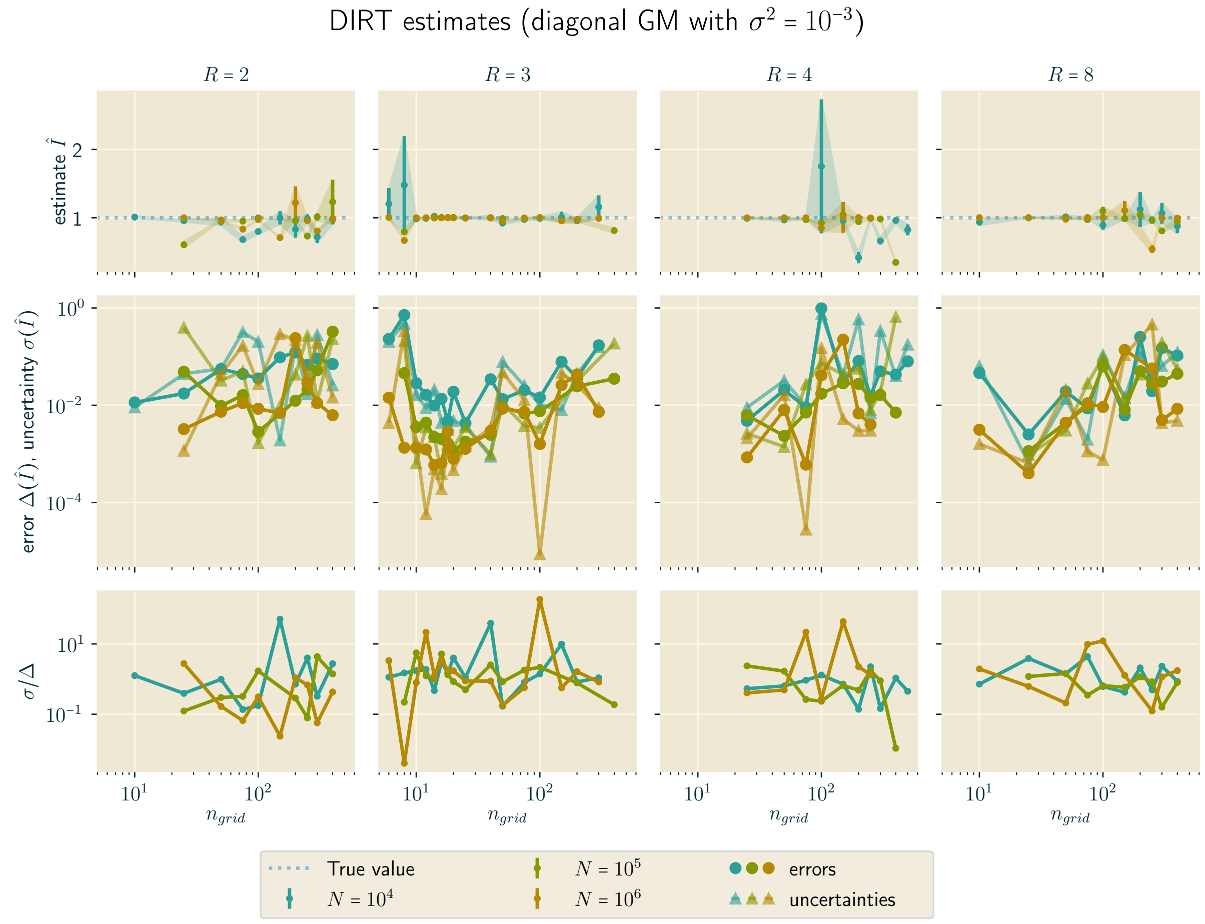

Figure 4.1: Varying the number of grid points n for different sets of (R_0, N).

As a start I vary the number of grid points n for different values of R_0 (initial tensor-train rank) and N (sample size). Assuming that the number of grid points directly translates to accuracy, my initial test runs aim at relatively high numbers of grid points[1]. The results can be seen in figure 4.1. It becomes apparent that increasing n does not lead to an improvement in accuracy. The plots in figure 4.1 and 4.2 give slight indication that increasing R stabilizes the accuracy of uncertainty estimates σ/Δ. However, this effect seems to be of minor importance and n to be the more impactful parameter.

At first glance it is puzzling that large parameters n lead to a decrease in accuracy. Running DIRT for grid parameters $n \in [0, 25]$ shows that the algorithm works best for $n \sim 20$. Facing smaller grid sizes the result and it's accuracy also get significantly worse. For $n < 6$ DIRT is not able to provide an estimate at all. A possible reason for the loss of accuracy with $n \to \infty, n > 20$ might be that overly refined grids introduce too much noise into the approximating functions, rendering approximations more difficult. This proposed explanation is supported by a look at the curve described by the Hellinger error, shown in figure 4.3. In this plot it can be observed that D_H becomes optimal for the pairs $(R = 3, n \sim 20)$ and $(R = 12, n \sim 30)$. Hence, I deduce that

[1]In their paper on DIRT Cui et al. typically vary n in the range of 10 to 30, N in the range of 10^3 ti 10^6 and R only for small values[3]. However, I decided out of curiosity to perform the tests within a larger parameter space

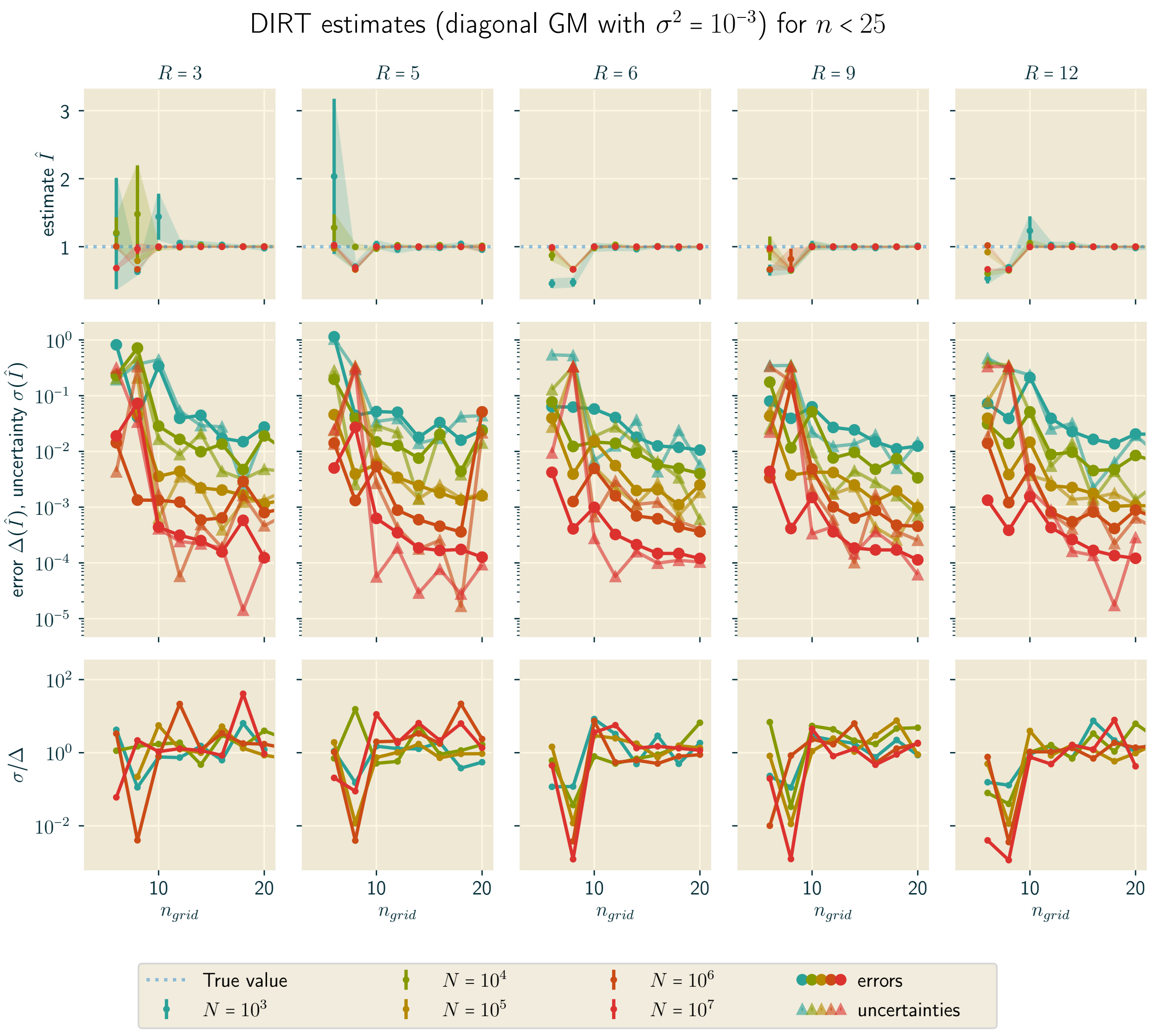

Figure 4.2: Varying the grid resolution parameter n for smaller values and different combinations of (R_0, N).

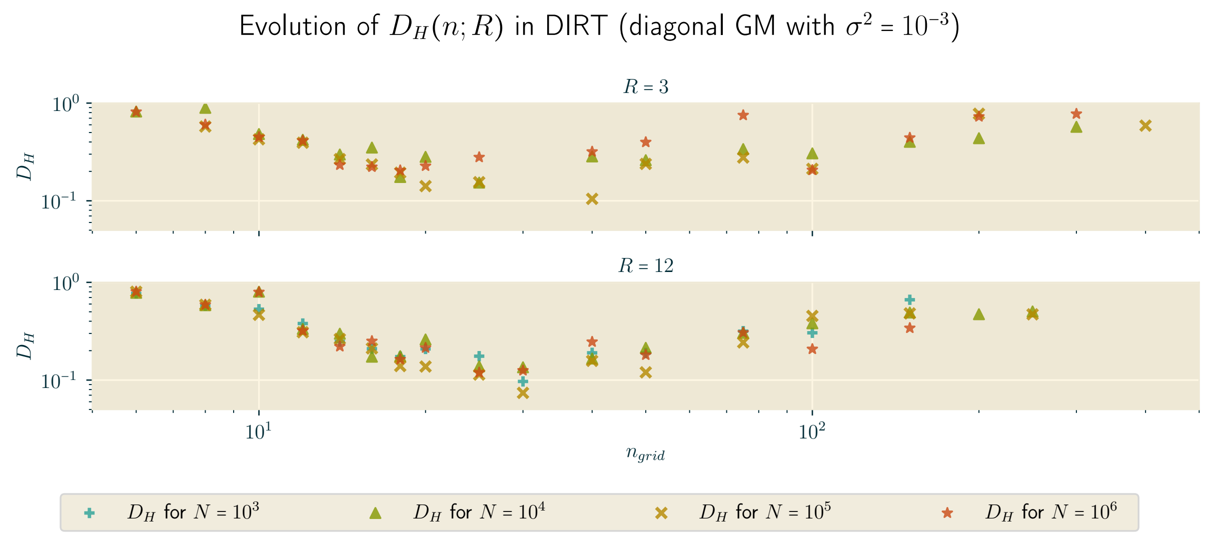

Figure 4.3: Hellinger error D_H for the approximations to the diagonal GM $g(\vec{x};\vec{\mu}_i,\sigma^2 = 10^{-3})$ constructed by DIRT.

with growing R the position of the optimum in n increases and the Hellinger error at optimal n decreases. Interestingly, the impact of R seems to be limited to the region surrounding the optimum.

Varying the sample size N, the naïve expectation is that the accuracy scales with N. As figure 4.4 shows, this indeed applies. Especially for $R_0 = 15, 16, 18$ the uncertainty approximately follows $\log\sigma \sim -\frac{1}{2}\log N$. Compared to VEGAS, I conclude that DIRT typically yields better results for $N < 10^5$. Above that boundary not even optimized parameters allow DIRT to compete with VEGAS.

Taking all runs into account, DIRT seems to give slightly more reliable uncertainty estimates than VEGAS (typically $0.1 \leq \sigma/\Delta \leq 10$) . Furthermore, if VEGAS misses a peak, it is not able to become aware of this, thus underestimating the uncertainty drastically (c.f. fig. 4.5).

In total increasing N by a factor of 10^4 allows DIRT to improve the actual deviation from the correct result Δ from $\sim 10^{-1}$ to $\sim 10^{-3} - 10^{-4}$, whereas VEGAS manages to improve from $\sim 10^0$ to 10^{-4}. This huge improvement is mainly due to the fact that VEGAS performs better than DIRT for $N \geq 10^5$, but for lower sample sizes significantly worse.

So far I only tested DIRT for its optimal configuration in the case of $\sigma^2 = 10^{-3}$. In different runs I also tested VEGAS for varying combinations of iterations n_{itn} and preconditioning iterations $n_{itn,pre}$. The results indicate for this specific setting that preconditioning has no significant influence on the quality of the estimate. Especially for $N \geq 10^5$ the quality does not depend on the number of preconditioning or main iterations. Above this boundary VEGAS is able to identify all peaks and the computed uncertainties also match the actual deviations. Overall, the plots in figure 4.5 reveal that for $N \geq 10^5$ the relative improvements in accuracy achieved by VEGAS are similar to DIRT.

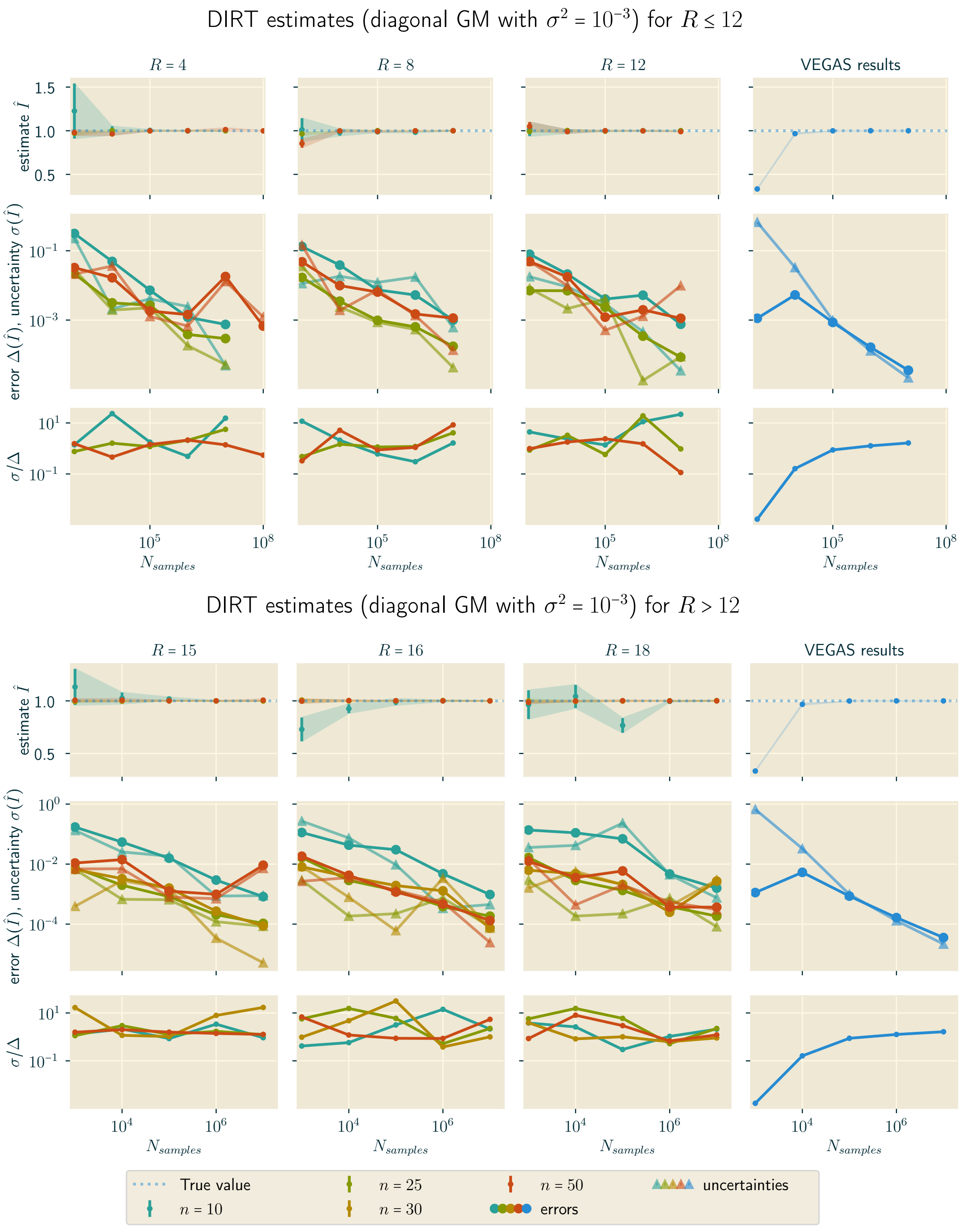

Figure 4.4: Varying the number of samples N for different numbers of initial TT-ranks. The VEGAS result provided as a comparison is identical for both subfigures. The shown VEGAS estimates stem from a VEGAS run with $n_{itn} = 25$ and a preconditioning phase of $n_{itn,pre} = 10$, one of the best runs achieved with VEGAS.

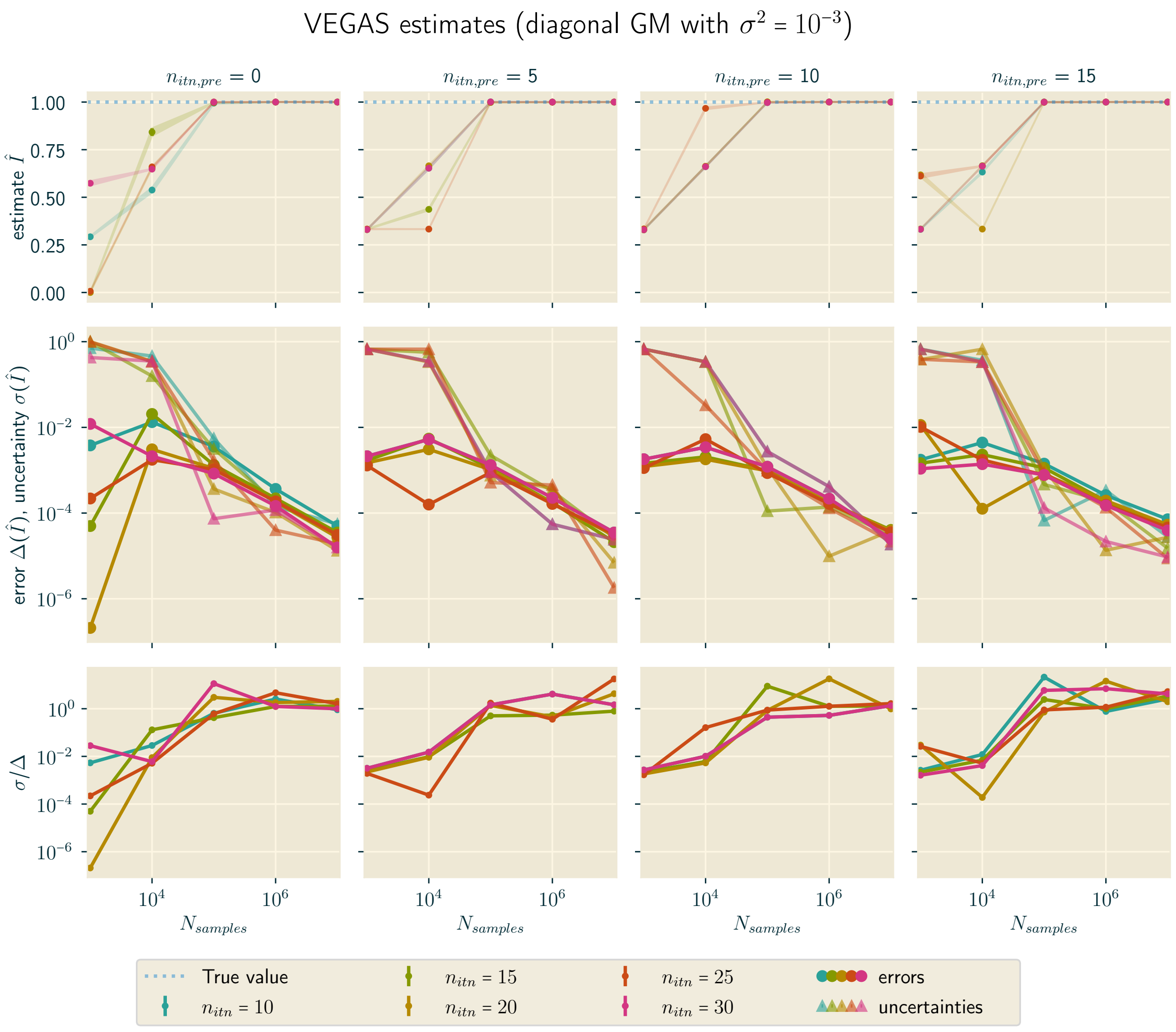

Figure 4.5: VEGAS estimates for different configurations.

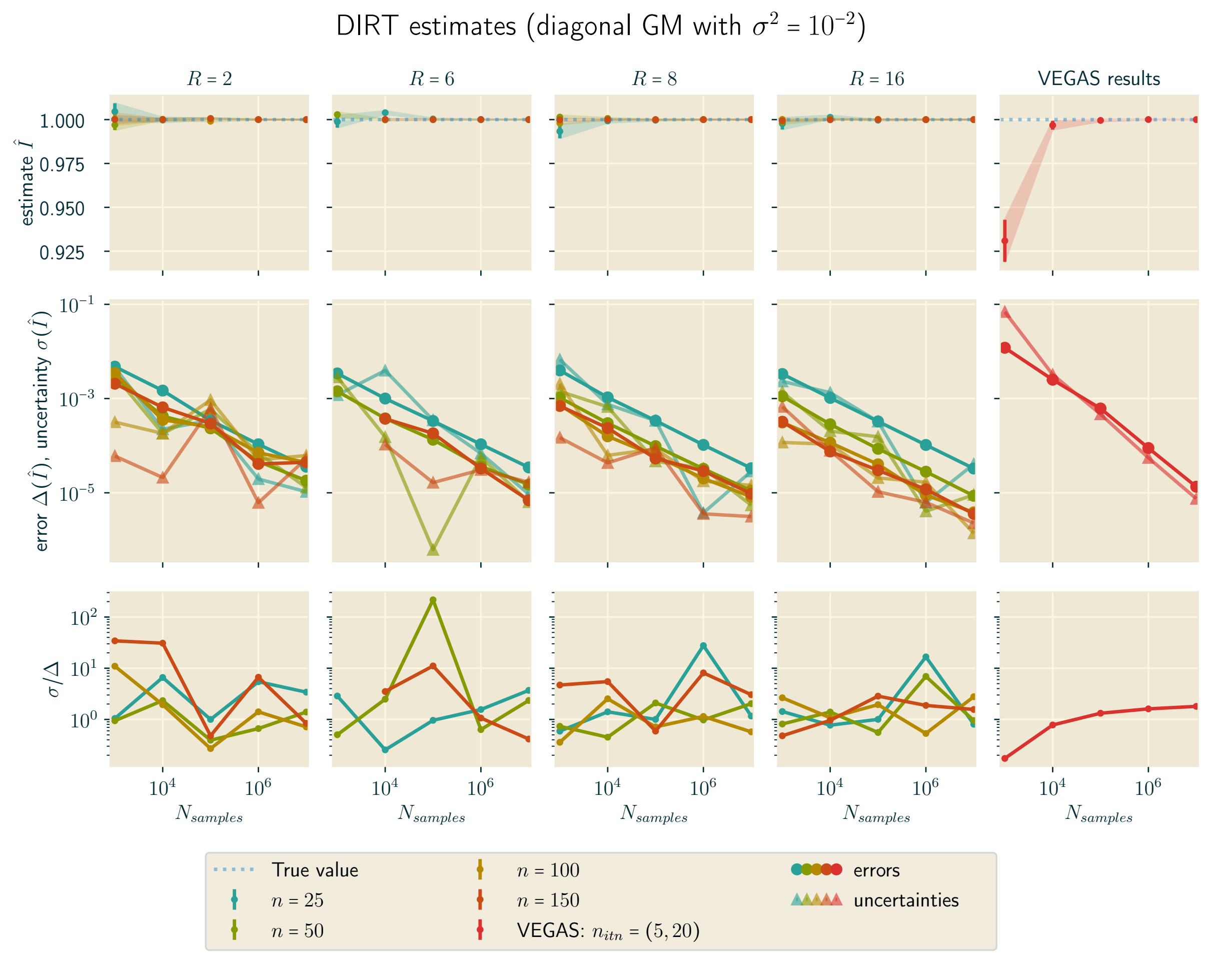

Figure 4.6: Varying the number of samples N for different numbers of initial TT-ranks R_0 and $g(\vec{x}; \vec{\mu}_i, \sigma^2 = 10^{-2})$.

4.1.2 Diagonal GM with σ^2=10^{-2}, 10^{-4}

Taking a look at broader and smaller peaked Gaussian mixtures reveals further information. The case of $\sigma^2 = 10^{-2}$ allows to indicate whether DIRT can also be used for less narrow structures, where VEGAS is expected to perform better. On the other hand, a look at $\sigma^2 = 10^{-4}$ allows predictions whether DIRT is better capable of resolving narrow features than VEGAS.

GM with σ^2=10^{-2}

For $N \le 10^6$, DIRT does perform better than VEGAS, independent from the parameter settings. In the most optimal cases, $R_0 = 16$ and $n = 100, 150$, the results obtained by DIRT are $2 - 50\times$ as accurate as the results obtained by VEGAS for 5 preconditioning and 20 main iterations (c.f. fig 4.6). Comparing the results shown in figure 4.6 and 4.8 allows to deduce that VEGAS is using samples more efficiently in the case of larger sample sizes ($N \ge 5$), as $\log {}^{\sigma(N_2)}/_{\sigma(N_1)} \sim -\log {}^{N_2}/_{N_1}$. For $N \le 10^5$, it performs similar to DIRT, with $\log {}^{\sigma(N_2)}/_{\sigma(N_1)} \sim -0.5 \log {}^{N_2}/_{N_1}$.

A further look at figure 4.6 allows to confirm what has been observed for the case of a GM

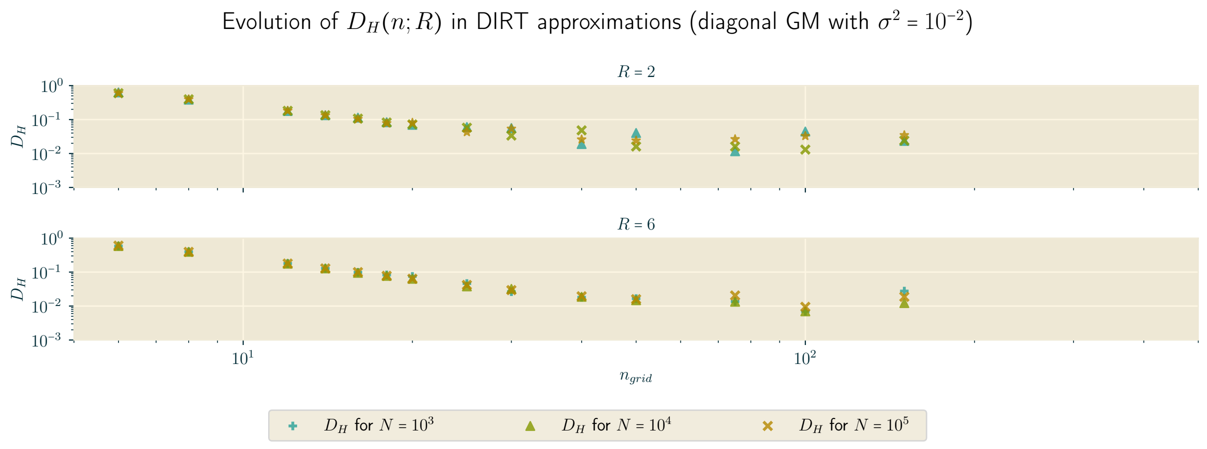

Figure 4.7: The Hellinger error in the DIRT approximations of $g(\vec{x}; \vec{\mu}_i, \sigma^2 = 10^{-2})$ exhibits a similar behaviour as for $g(\vec{x}; \vec{\mu}_i, \sigma^2 = 10^{-3})$. The position of the optimum however is shifted. Additionally, the approximation errors are smaller for most n, compared to figure 4.3.

with $\sigma^2 = 10^{-3}$ before: increasing R_0 allows larger grid sizes n to yield more accurate results. This is also confirmed by the graph described by the Hellinger errors in figure 4.7. The most apparent feature of this plot is, that the Hellinger approximation errors are up to ten times smaller then in the $\sigma^2 = 10^{-3}$ case. This however is less surprising considering the fact that, for less narrow features, the integrand can be easier approximated. Furthermore, it is in accordance with the computed estimate uncertainties and actual deviations (shown in figure 4.6), which face an improvement up to ten times compared to the $\sigma^2 = 10^{-3}$ case. Notable is that the optimum grid parameter n lies now in the range of $\sim 75 - 100$. The Hellinger errors for small $n \leq 20$ however reside in similar regimes as before. This allows an important conclusion: improvements in accuracy can be achieved by either increasing N or by increasing both n and R. It seems that the latter option is the more effective.

Identical to the previous case, preconditioning does not have a significant effect on the results determined by VEGAS. Surprisingly, it still faces problems to identify all peaks for $N \sim 10^3$. For $N \geq 10^4$, VEGAS is able to provide reliable estimates of the integral. The uncertainty estimates provided by VEGAS with 5 or more preconditioning iterations prove more reliable than for DIRT.

GM with σ^2=10^{-4}

This case proves to be most challenging for both VEGAS and DIRT so far. As illustrated by figures 4.9, 4.11, 4.10 and 4.12, both algorithms fail to provide correct results in most of the cases.

Using DIRT with the standard tempering of $\log \beta_k = -4 + k$, the best results are achieved for $R_0 = 12, n = 20$. However, even with this set of parameters, DIRT has difficulties to identify the correct shape of the integrand. The returned uncertainty estimates are typically to small by a

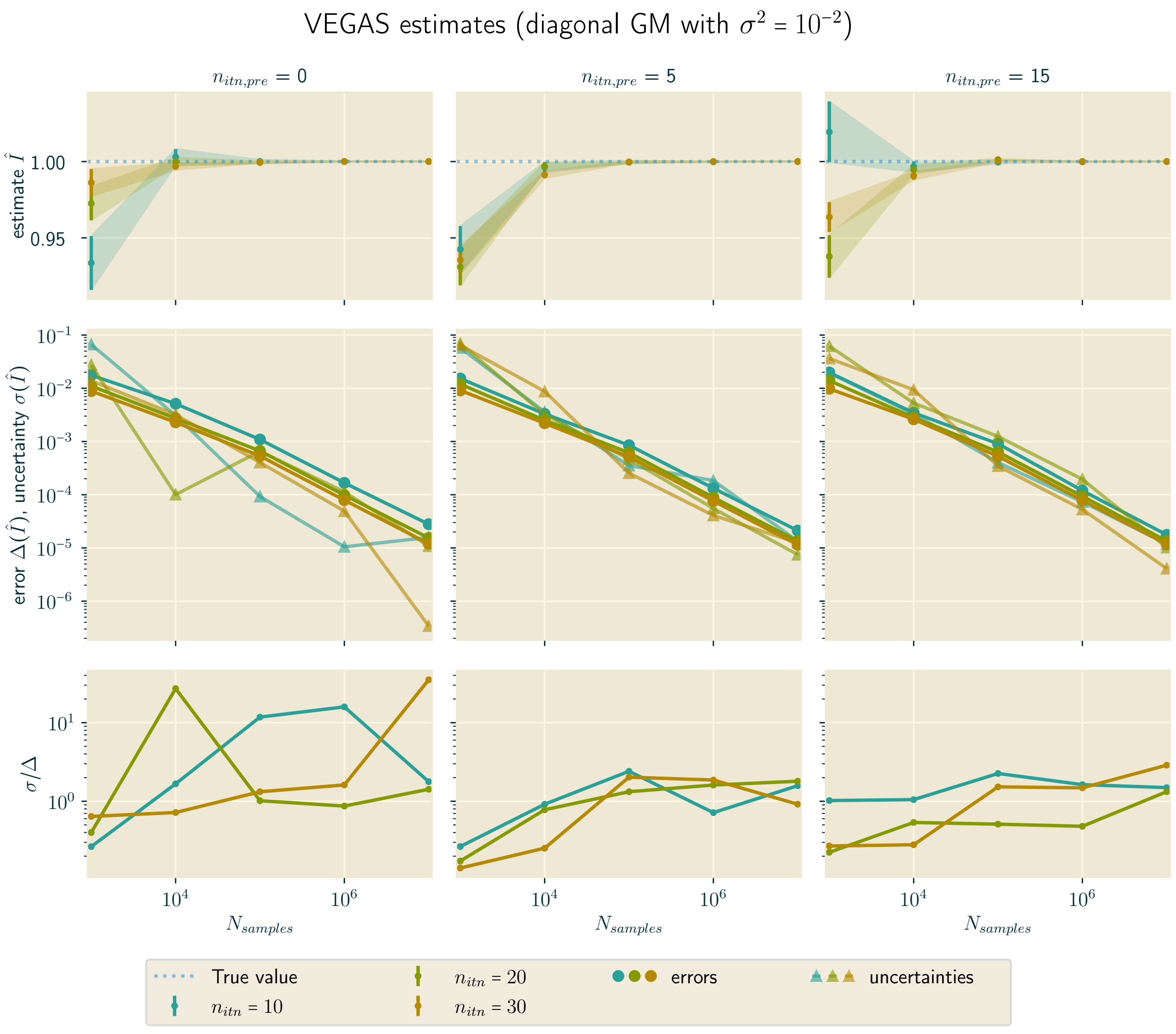

Figure 4.8: VEGAS estimates for different configurations in the case of $g(\vec{x};\vec{\mu}_i,\sigma^2 = 10^{-2})$.

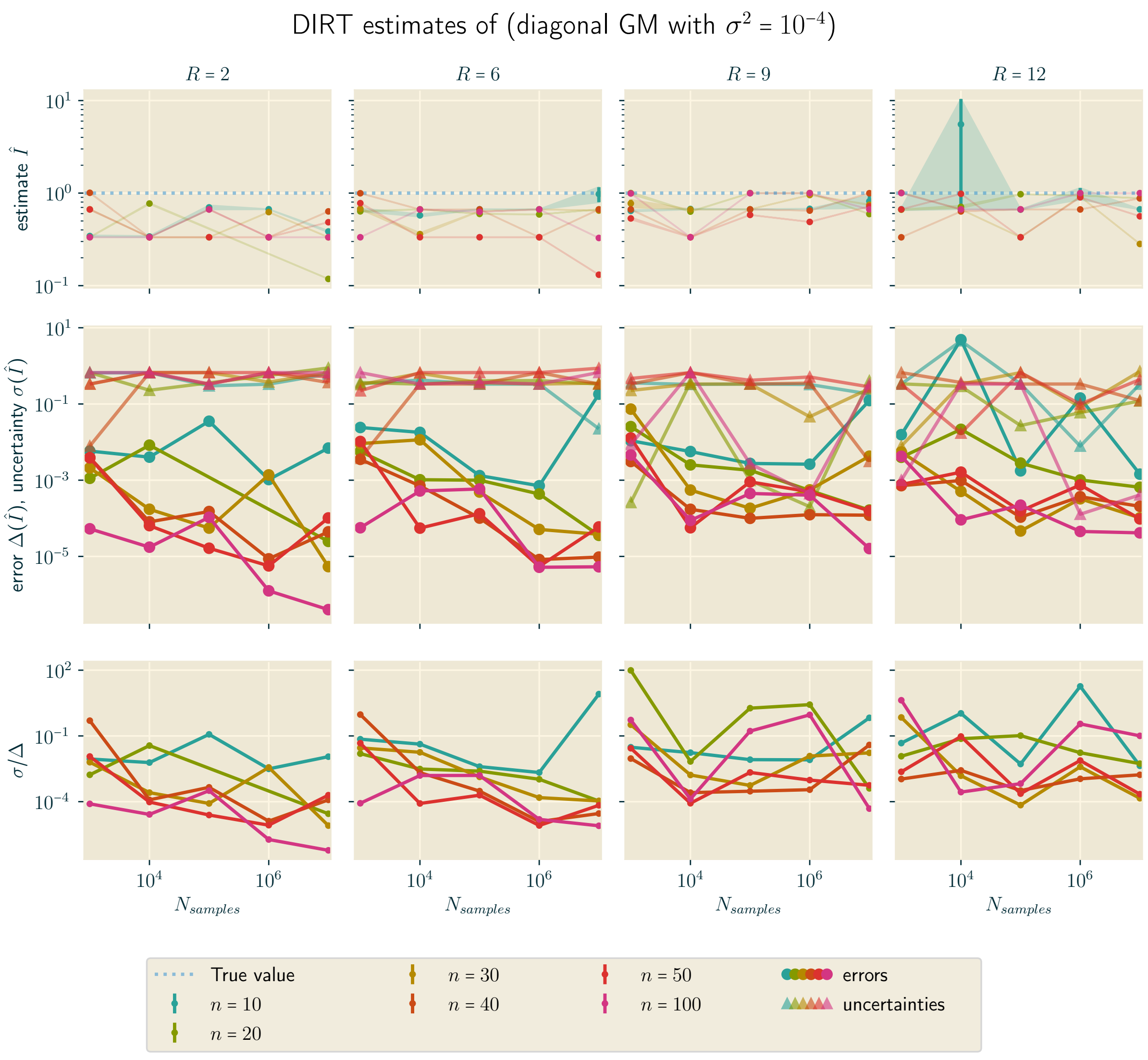

Figure 4.9: Varying the number of samples N for different numbers of initial TT-ranks R_0 and $g(\vec{x}; \vec{\mu}_i, \sigma^2 = 10^{-4})$.

factor of 10 – 100. Figure 4.9 slightly indicates that increasing R_0 also increases the probability of peak identification, as for $R = 9, 12$, typically two peaks can be identified.

Expecting that a slower tempering might enable DIRT to adapt to the integrand more easily, I first change the step size, so that $\log \beta_k = -4 + k/2$. This refined tempering leads to minor improvements. For $R = 9, n = 25, N < 10^6$ and $R = 12, n = 20, N < 10^6$, correct results and satisfying uncertainty estimates can be provided. For $R = 3, 6$ the results quality decreased, compared to the standard tempering (c.f. figure 4.10).

As a last attempt, I run DIRT with an even more refined tempering, setting $\mathrm{ld}\beta_k = -13 + k$. The results are shown in figure 4.11. Again, this yields no significant changes. Only one set of parameters, $R = 12, n = 25$, is capable of providing correct estimates for $\log N = 3, 4, 5, 6$.

VEGAS, on the other hand, fails consistently to provide correct results for $N < 10^5$, but does

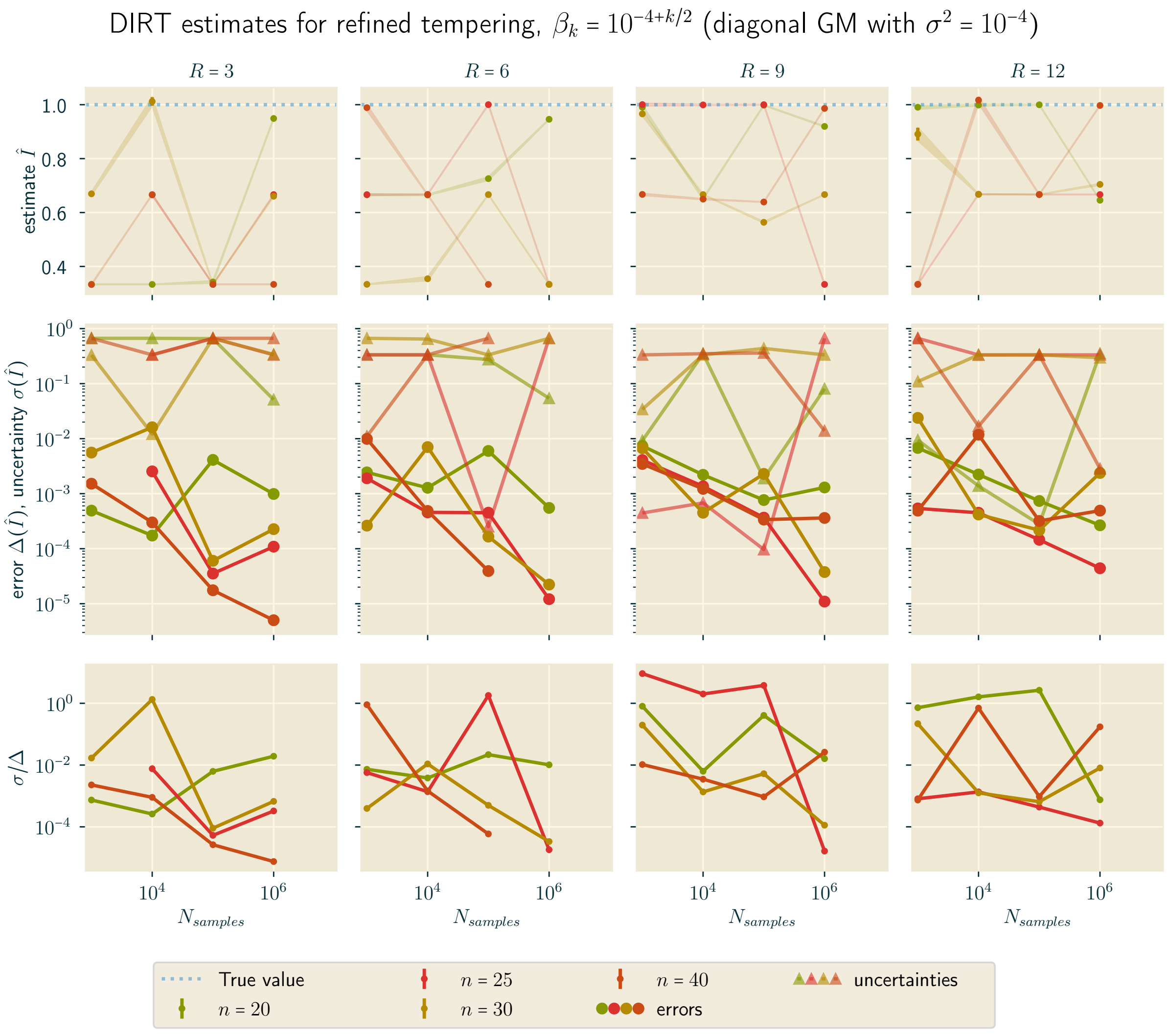

Figure 4.10: Varying the number of samples N for different numbers of initial TT-ranks R_0 and $g(\vec{x}; \vec{\mu}_i, \sigma^2 = 10^{-4})$. Note that for $R = 3, n = 25$, the data point at $N = 10^3$ has been removed. It deviated by a factor of $\sim 10^6$.

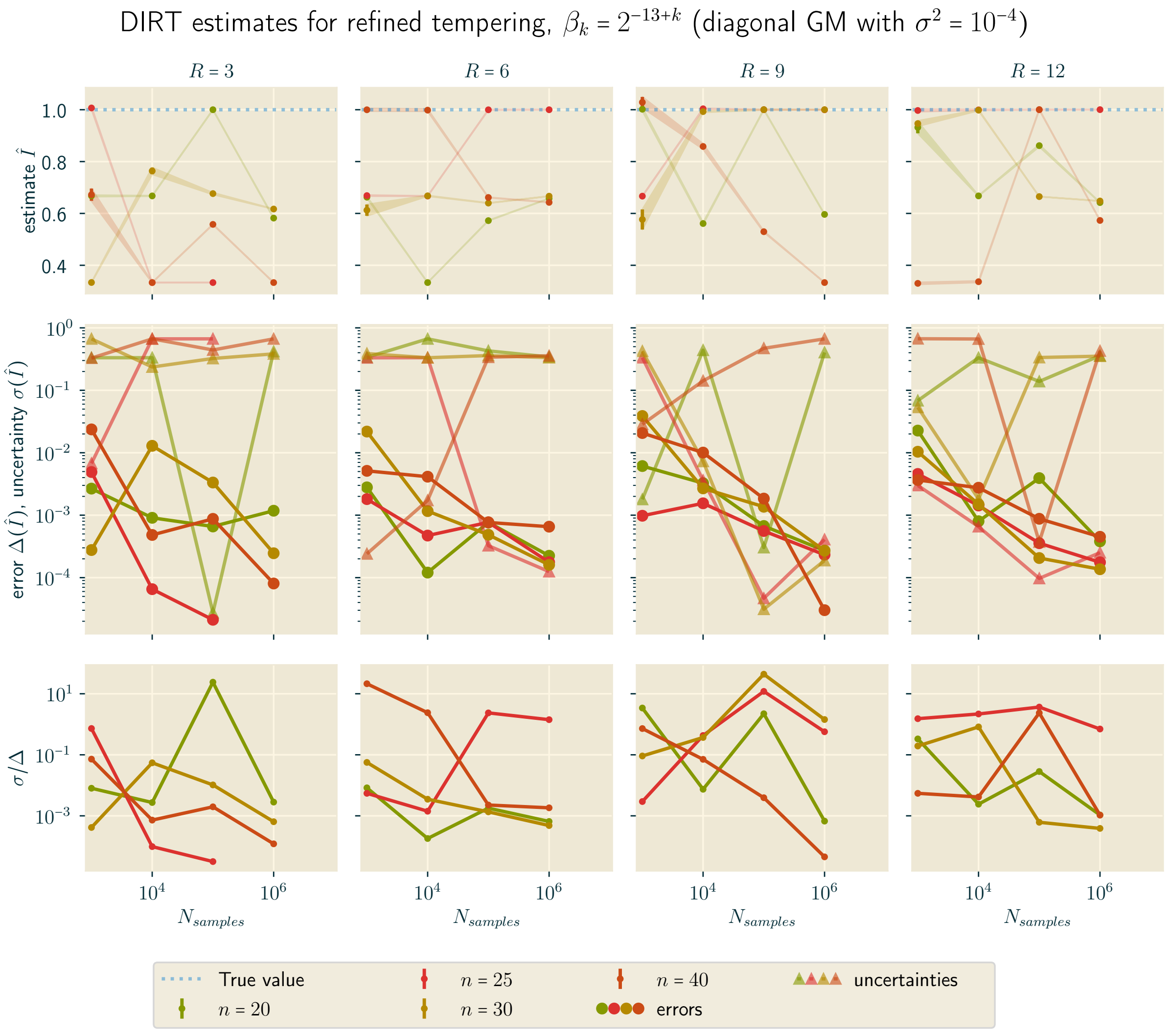

Figure 4.11: Varying the number of samples N for different numbers of initial TT-ranks R_0 and $g(\vec{x};\vec{\mu}_i,\sigma^2 = 10^{-4})$. Note that for $R = 3, n = 25$, the data points at $N = 10^6, 10^7$ have been removed. They deviated by a factor of ~ 10^6.

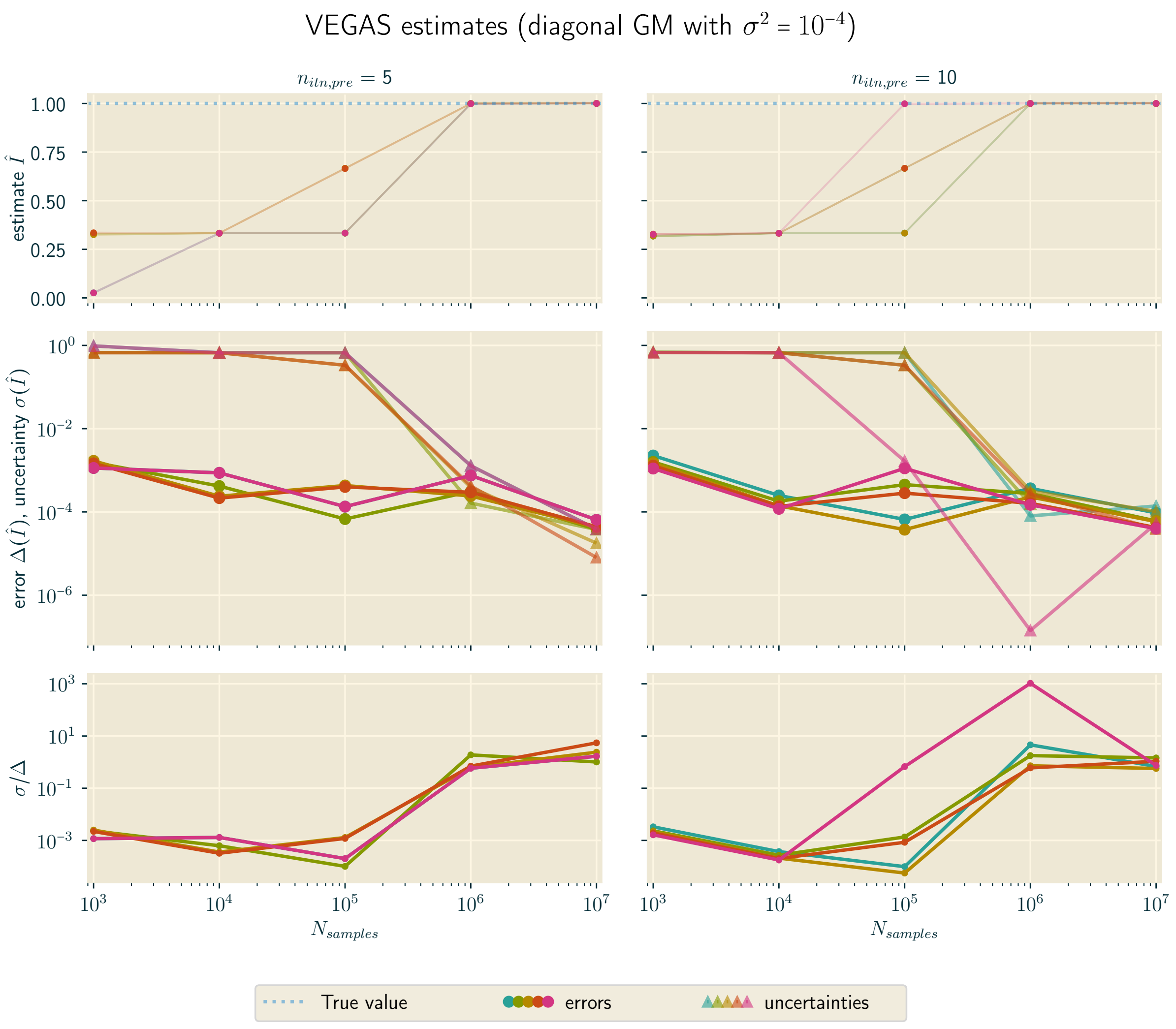

Figure 4.12: VEGAS estimates for different configurations in the case of $g(\vec{x};\vec{\mu}_i,\sigma^2 = 10^{-4})$.

yield precise estimates for $N \geq 10^6$. The accuracies of these results are better than the best ones provided by DIRT.

4.2 Non-Diagonal Gaussian Mixture

To ensure that the previous results were not influenced by the diagonal alignment of the peaks, I let DIRT and VEGAS estimate the integral value of a Gaussian mixture, similar to equation (2.19), but with $\sigma^2 = 10^-3$ and

$$\mu_1 = \left(-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}\right)^\intercal , \ \mu_2 = \left(0, 0, \frac{1}{2}, \frac{1}{2}\right)^\intercal , \ \mu_3 = \left(\frac{1}{2}, \frac{1}{2}, 0, 0\right)^\intercal .$$

The results, shown in figure 4.13, do not give any reason to suspect that DIRT performs differently than for diagonally aligned peaks. However, they indicate again that DIRT performs better than VEGAS for small sample sizes, $N \leq 10^5$.

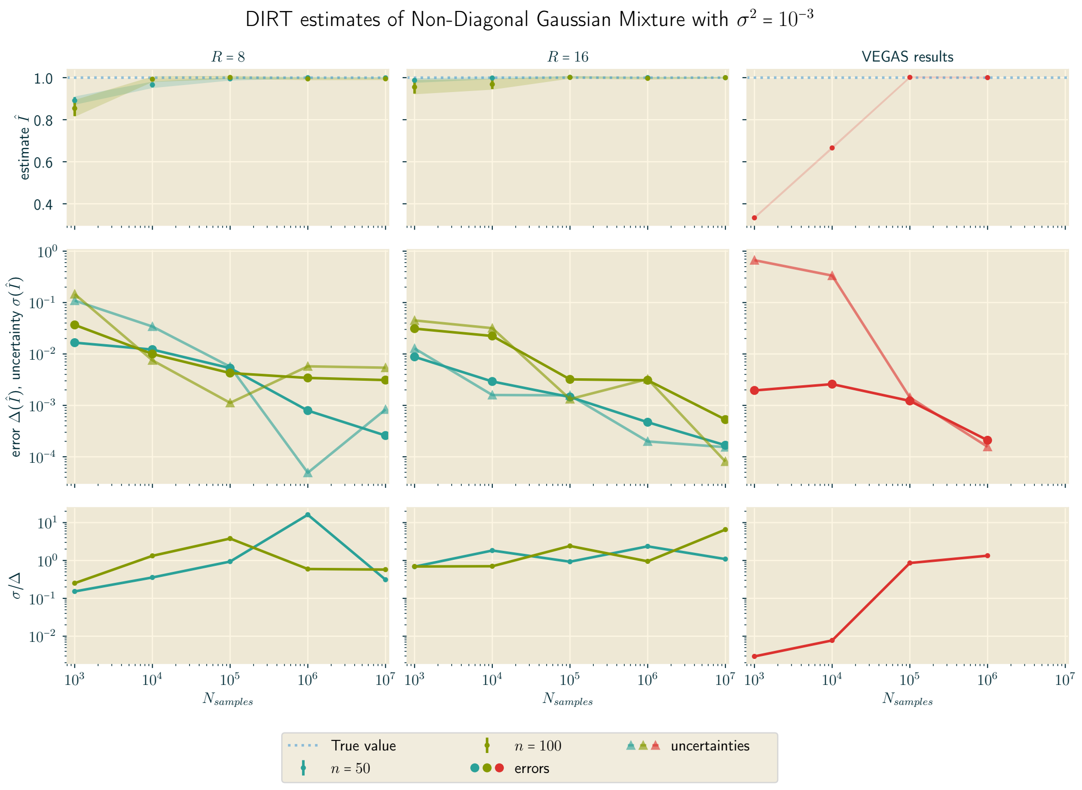

Figure 4.13: Varying the number of samples N for different numbers of initial TT-ranks R_0.

4.3 Case study: Using DIRT to constrain new Physics

As discussed earlier, VEGAS requires a proper treatment to correctly solve equation (2.49). Astonishingly, DIRT faced huge issues creating a proper mapping for it, even when the integration domain was constrained to the peak region. Based on the code provided in [17, 11], I developed three versions of the integrand in MATLAB. The first implementation followed Python code as closely as possible. Trying to adapt a less Pythonic way of coding, I wrote a second version. The last edition discarded the substitution $\cos\theta \to y$, thereby allowing DIRT to deal with fixed integration boundaries, but on the cost of more difficult peak resolution.

Of all versions, only the second approach allowed DIRT to return estimates in some distinct cases, i.e. for $g = 1$, $m_{Z'} = 1, 5, 10, 100, 1000\,\text{MeV}$ in the case of annihilating electrons. For these runs, I chose $R_0 = 20$, $n = 50$, $N = 10^4, 10^5$ and a very cautious tempering, $\text{ld}\beta_k = -4 + {}^k\!/\!_2$, in some cases even $\text{ld}\beta_k = -4 + {}^k\!/\!_4$. These parameters allowed to run DIRT also for smaller values of g (I tested up to $g = 10^3$). However, these runs were typically aborted at steps equivalent to temperings of $\beta \sim 0.6 - 0.7$ for too bad approximation quality.

For other settings of the physical parameters, AMEN (the TT approximation algorithm implemented in DIRT) usually faced the issue of (close-to) singular matrices, not allowing to proceed further[2]. It is unclear why this issue occurred. However, it might be related to MATLAB not

[2]In fact, AMEN only raises a warning if (close-to) singular matrices occur. For the physics case study, however, the algorithm seemed to get stuck at this point.

providing support for quadruple precision floats, thus experiencing problems when dealing with both extremely small and large numbers. A further reason might be connected to the shape of the integration domain and the chosen substitutions. In the end, the created data did not allow further analysis.

5 Summary and Outlook

Contributing to the search for efficient numerical integration algorithms, this thesis investigated the applicability of DIRT. VEGAS serves as a benchmark model. Four-dimensional Gaussian mixtures with variances of $\sigma^2 = 10^{-2}$ (wide case), $\sigma^2 = 10^{-3}$ (medium case) and $\sigma^2 = 10^{-4}$ (fine case) were used as target functions.

The results confirm that DIRT is capable of providing an efficient integration algorithm. The initial assumption that DIRT could enhance fine resolution structure approximation through inherent tempering was not fulfilled. In fact, for narrow peaks and more complex structures, DIRT proved to be inadequate, while VEGAS could be optimised to compute correct results. Regarding future research involving DIRT, the potential for improvements through a change of basis functions should be considered, as this aspect remained unexplored in this thesis.

When dealing with medium-sized and wide peaks ($\sigma^2 = 10^{-2}, 10^{-3}$) in Gaussian mixtures, DIRT's performance increased. For sample sizes $N \lesssim 10^6$ it produced similar or even more accurate results than VEGAS. When tested on a Gaussian mixture with $\sigma^2 = 10^{-2}$, this was independent of the initial choice of parameters. However, the combined optimisation of grid size and initial TT rank allowed DIRT to achieve same precision results while utilising only 10 to 100 times fewer samples than VEGAS.

Regarding the physical case study, DIRT failed to deliver. It could not be explored whether this failure originated in the implementation of the integrand in MATLAB, MATLAB imminent issues or an actual issue of DIRT, e.g. with complicated integration domains.

For the future, an implementation of DIRT in Python seems desirable. Such a project could use the Python IRT version already published by Dolgov et al. [6, 3]. However, it should be noted that the required TT library `TT-toolbox` relies on Fortran code, which has been experienced to be challenging to use under Windows operating systems [13]. Alternatively, the `torchTT` library may be a more promising solution, as it is compatible with `PyTorch` [8].

Overall, DIRT appears to be a practical sampling technique for Monte Carlo integration, possibly more fitting for typical integration problems with limited sample sizes. In such instances, it is anticipated to have a similar or superior performance to VEGAS.

6 Bibliography

For translations and formulation, I used DeepL [16].

[1] D. P. Aguillard et al. *Measurement of the Positive Muon Anomalous Magnetic Moment to 0.20 ppm.* 2023. DOI: https://doi.org/10.48550/arXiv.2308.06230. arXiv: 2308.06230 [hep-ex].

[2] Robert Bollig et al. "Muons in Supernovae: Implications for the Axion-Muon Coupling". In: *Phys. Rev. Lett.* 125 (5 July 2020), p. 051104. DOI: 10.1103/PhysRevLett.125.051104. URL: https://link.aps.org/doi/10.1103/PhysRevLett.125.051104.

[3] Tiangang Cui and Sergey Dolgov. "Deep Composition of Tensor-Trains Using Squared Inverse Rosenblatt Transports". In: *Foundations of Computational Mathematics* 22.6 (July 2021), pp. 1863–1922. DOI: 10.1007/s10208-021-09537-5. URL: https://doi.org/10.1007%2Fs10208-021-09537-5.

[4] DeepTransport. *TT-rare.* 2023. URL: https://github.com/DeepTransport/TT-rare.

[5] Sergey Dolgov et al. *Approximation and sampling of multivariate probability distributions in the tensor train decomposition.* 2019. arXiv: 1810.01212 [math.NA].

[6] Sergey Dolgov et al. *TT-IRT.* 2022. URL: https://github.com/dolgov/TT-IRT.

[7] Charles R. Harris et al. "Array programming with NumPy"". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[8] Ion Gabriel Ion. *Torch TT.* 2023. URL: https://github.com/ion-g-ion/torchTT.

[9] G. Peter Lepage. "A new algorithm for adaptive multidimensional integration". In: *Journal of Computational Physics* 27.2 (1978), pp. 192–203. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(78)90004-9. URL: https://www.sciencedirect.com/science/article/pii/0021999178900049.

[10] G. Peter Lepage. "Adaptive multidimensional integration: vegas enhanced". In: *Journal of Computational Physics* 439 (Aug. 2021), p. 110386. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2021.110386. URL: https://doi.org/10.1016%2Fj.jcp.2021.110386.

[11] Claudio Andrea Manzari et al. "Supernova Limits on Muonic Dark Forces". In: (July 2023). arXiv: 2307.03143 [hep-ph].

[12] I. V. Oseledets. “Tensor-Train Decomposition”. In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317. doi: 10.1137/090752286. eprint: https://doi.org/10.1137/090752286. url: https://doi.org/10.1137/090752286.

[13] Ivan Oseledets et al. *TT-Toolbox*. 2023. url: https://github.com/oseledets/TT-Toolbox.

[14] Planck Collaboration et al. “Planck 2018 results - VI. Cosmological parameters”. In: *A&A* 641 (2020), A6. doi: 10.1051/0004-6361/201833910. url: https://doi.org/10.1051/0004-6361/201833910.

[15] Murray Rosenblatt. “Remarks on a Multivariate Transformation”. In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 470–472. doi: 10.1214/aoms/1177729394. url: https://doi.org/10.1214/aoms/1177729394.

[16] DeepL SE. *DeepL*. Version 4.9.0.10395. 5th Sept. 2023. url: https://deepl.com/translator.

[17] spinjo. *SNforMuTau*. 2023. url: https://github.com/spinjo/SNforMuTau.

[18] Eleonora Di Valentino et al. “In the realm of the Hubble tension—a review of solutions*”. In: *Classical and Quantum Gravity* 38.15 (July 2021), p. 153001. doi: 10.1088/1361-6382/ac086d. url: https://doi.org/10.1088/1361-6382/ac086d.

7 Appendix

7.1 MATLAB Code: MC integration with DIRT

The following code provides a minimal example of an integration algorithm utilising DIRT for sampling.

```
% load dependencies from tt-toolbox
addpath('../tt-irt-matlab/constructors');
addpath('../tt-irt-matlab/samplers');
addpath(genpath('../tt-irt-matlab/TT-Toolbox-small'));
addpath('../tt-irt-matlab/utils');

% set dimension and domain
d = 4;
[a,b] = deal(-1, 1);

% set sample size
Nsamples = 1e4;

% set grid size and create grid
ngrid = 20;
x = linspace(a, b, ngrid)';
xsf = repmat({x}, d, 1);

% set initial TT-rank
R0 = 1;

% set tempering powers
[beta_min, beta_step, beta_max] = deal(-4, 1, 0);
beta = 10.^(beta_min:beta_step:beta_max);

% DIRT additional parameters
boundary = true;
```

```
reference = 'n1.5';

% define logarithm of target function
lfun = @(x,beta_km1,beta_k)log(fun(x,args).*(beta_k-beta_km1);

% measure time with tic --- toc
tic
try
% Build DIRT
IRT = tt_dirt_approx(xsf, lfun, beta, 'vec', true, 'y0', R0, '
    boundary', boundary, 'reference', reference, 'plotdiag',
    false);
[z, lFapp] = tt_dirt_sample(IRT, randref(IRT.reference, Nsamples
    , d));
true_vals = lfun(z,0,1);

% get estimates from MC integration
z = double(z)';
lFapp = double(lFapp)';
lFex = double(lfun(z, 0, 1));

mc_est = mean(exp(lFex - lFapp));
mc_err = sqrt((mean( exp(lFex-lFapp).^2) - mc_est^2) / length(
    lFex));

fprintf('MC result: %f +/- %f \n', mc_est, mc_err);

toc
catch ME % catch error messages
toc
disp(ME.message);
end
```

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 05.09.2023,

[illegible]

Noah Kriesch