GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Bachelor's Thesis

Institut für Theoretische Physik

Using Hamiltonian Monte Carlo Techniques for Phase Space Sampling

Hamilton Monte Carlo Methoden zum Phasenraum Sampling

prepared by

Mathis Gerdes

from Göttingen

Abgabedatum:	22. Juni 2018
Erstgutachter:	Prof. Dr. Steffen Schumann
Zweitgutachter:	Prof. Dr. Marcus Müller

Abstract

The generation of phase space events in high energy particle physics is commonly done using a Metropolis-Hasting unweighting algorithm with fixed proposal distribution. The multi-channel Markov chain Monte Carlo algorithm (MC^3) introduced in [1] proposes a mixing of the fixed proposal method with a local Metropolis update. Based on this framework, the feasibility and performance improvement of using Hamiltonian (Hybrid) Monte Carlo (HMC) is analyzed. Performance is measured based on the number of target density evaluations and the introduced autocorrelation between events. Based on the analysis of a toy problem, the combined HMC-MC^3 is found to improve on the sample convergence behavior of the fixed target proposal and reduces the autocorrelation of the local HMC method for strongly peaked distributions. While the overall approach seems promising, various parameters must be tuned for individual target distributions to obtain optimal performance.

Contents

1. Introduction

The physics of high energy particle collisions is fundamentally described by a matrix element $\mathcal{M}$, a function of the initial and final state four-momenta of the particles, which is proportional to the transition amplitude. Generally, one is interested in finding an expectation value $\langle \mathcal{O} \rangle$ of some observable with respect to this process, given by a $3n-4$ dimensional [2] integral (where n is the number of final state particles, $n \geq 2$) of the form

$$\langle \mathcal{O} \rangle = \frac{\int \mathrm{d}p\, \mathcal{O}(p) f(p)}{\int \mathrm{d}p\, f(p)}, \tag{1.1}$$

where $f \propto |\mathcal{M}|^2$ is the differential cross section. Because of the relatively high dimensionality it is natural to turn to Monte Carlo integration, for which the convergence behavior is independent of the dimension [3]. Furthermore, using importance sampling introduced in section 2.4, Monte Carlo integration can be extended to better handle strongly peaked integrands as are common in this field. The model of the physical process is split into three regimes of momentum-transfer (referred to as factorization), which are simulated individually: a first phase of few, high energetic partons, parton showering (commonly simulated via empirical models), and a final hadronization into observable particles [2]. Subject of this report is the first regime, for which a perturbative hard matrix element $\mathcal{M}$ can be algebraically derived by a computer. In general, to make predictions about physical observations in a detector, the three fundamental phases of simulation are followed by a detector simulation.

The objective of the first simulation phase is to compute the total cross section $\sigma = \int \mathrm{d}p\, f(p)$ and generate samples (points in phase space corresponding to, and sometimes referred to as, events) distributed according to f. Each generated point in phase space may generally have a corresponding weight such that the expectation value $\langle \mathcal{O} \rangle$ is approximated by the weighted mean of $\mathcal{O}$ evaluated at the sample points (see section 2 on Monte Carlo integration). If the weights for all sample points are equal, the sample is said to be unweighted (i.e. the sample contains unweighted events). The samples generated in the first phase are propagated in the following two phases using phenomenological models [2], and finally input to a detector simulation. Since the detector simulation is generally computationally expensive, it is desirable to generate unweighted events. Furthermore, for the sample to be treatable as a substitute for experimental data it has to be unweighted, and must not display significant correlation between sample points. This is measured by the lag-autocorrelation defined in eq. (3.6). Since the matrix element is computationally expensive to evaluate, the number of function evaluations of f can be used as a heuristic for the computational cost of an algorithm.

In summary, the goal is to generate an unweighted phase space sample according to f with minimal lag-autocorrelation, while minimizing the number of function evaluations. This sample can be propagated using methods mentioned above, and can be used to compute $\langle \mathcal{O} \rangle$ for any observable of interest.

Given the constraints of unweighted and close to uncorrelated events, the established approach is to use Monte Carlo methods such as a Metropolis-Hasting Markov chain (section 3.2) with a fixed, global proposal distribution. While this approach ensures unweighted and uncorrelated events (apart from repetitions inherent to the algorithm; each next sample point is proposed independently of the previous one), distributions f with significant local peaks can lead to slow convergence to the target distribution. To avoid repetition of events in the final sample (and to counteract potentially slow convergence of the Markov chain), only every n^{th} sample point may be chosen as final event (this practice is referred to as subsampling [4]), effectively requiring multiple function evaluations per event. By using a local proposal mechanism the acceptance rate of the Markov chain update can be increased. However, the emergent random-walk behavior introduces significant correlation between sample points, and low-probability areas may not be crossed, leading to "undiscovered" and therefore underrepresented features of the distribution (in the limit the target distribution is reached but the convergence may be arbitrarily slow). The Multi-channel Markov chain Monte Carlo (MC^3) algorithm introduced in [1] (section 3.4) aims to mitigate this problem by combining a local Metropolis update (producing autocorrelated sample points) with a global update in analogy to acceptance-rejection sampling (section 3.1). Local "explorations" of the distribution are interrupted by global, random jumps in phase space that reduce the correlation and prevent the missing of features.

There are several variants of Metropolis methods, developed in other fields, that improve on the random walk behavior of a naive, symmetric, local update, using (local) information about the distribution (for example [5], [6], and [7]). After an overview of Monte Carlo integration and the basic sampling methods in sections 2 and 3, improvements on the basic MC^3 algorithm by using Hamilton Monte Carlo (HMC) [8] as local update are examined. It is shown that using the MC^3 algorithm in combination with HMC can significantly improve the sampling efficiency, but introduces the challenges of choosing good parameter values and having to deal with boundary conditions for certain distributions. Both of these issues, however, have known solutions such as the No-U-Turn sampler [6] and Sphercial HMC [9] which are variants of the HMC algorithm. Finally, the improved version of MC^3 is demonstrated on a simple differential cross section obtained via Sherpa [10].

Besides Sherpa all Monte Carlo methods are implemented, specifically for this examination, in Python and available at `https://github.com/mathisgerdes/hep-monte-carlo/releases/tag/v1.1`.

2. Monte Carlo Integration

Monte Carlo integration approximates an integral over a volume V by sampling the integrand at N (pseudo-) randomly selected points $x_i \in V$. In the simplest form of Monte Carlo integration (without any variance reducing techniques) the x_i are selected according to a uniform distribution in V, thus reducing the integral to an average:

$$I = \int \mathrm{d}x\, f(x) \approx E_N = \frac{V}{N} \sum_{n=1}^{N} f(x_n). \tag{2.1}$$

For N function evaluations, the statistical variance of the estimate E is [3]

$$\sigma^2(E) = \int_V \mathrm{d}x_1 \cdots \mathrm{d}x_N \left(E_N(x_1, \ldots, x_N) - I\right)^2 = \frac{\sigma^2(f)}{N}, \tag{2.2}$$

where

$$\sigma^2(f) = \int_V (f - I)^2 \tag{2.3}$$

is the variance of f on V. The Monte Carlo estimate of $\sigma^2(f)$ is

$$\sigma^2(f) \approx S^2(f) = \frac{V}{N} \sum_{n=1}^{N} f(x_n)^2 - E^2. \tag{2.4}$$

2.1. Statistical Distribution of Estimates

The central limit theorem implies that Monte Carlo estimates follow a normal distribution with variance according to eq. (2.2) and in the limit $N \to \infty$ the estimate E converges to the true value I [3]. It must be noted, however, that the above analytical results only hold if f is square-integrable. While in the limit the estimate for a none square-integrable integrand still converges to the true value, the error prediction is not reliable.

Fig. 2.1 shows the distribution of 2000 integration estimates for a square-integrable and a none square-integrable function

$$f(x, y) = (xy)^{-1/2} \qquad \text{and} \qquad f(x, y) = \sin(2\pi x) \cdot \sin(2\pi y) \tag{2.5}$$

respectively, over the unit cube $[0, 1]^2$ with the fixed number $N = 5000$ of function evaluations each run. In comparison, the normal distributions with variances according to eq. (2.2) (specifically the mean of such predictions over all runs) are shown. The distribution of the square-integrable integrand can be seen to closely resemble the normal distribution as predicted by the central limit theorem. The histogram for the none square-integrable function shows significant outliers and appears to be asymmetric.

A quantitative measure of how well the distributions match the predictions is given by a bin-wise χ^2 test, introduced in more detail in section 3.3.2, comparing the number of points in each bin to the predicted number of points according to the normal distribution. For M bins, the χ^2 statistic (eq. (3.7)) follows a χ^2 distribution with $M-1$ degrees of freedom (dof), if the the sample follows the expected (here normal) distribution. The mean of χ^2/dof is 1.

Using the bins as shown in fig. 2.1 ($M = 160$) it is $\chi^2/(M-1) = 0.9664$ for the square-integrable and $\chi^2/(M-1) = 1.123 \times 10^{15}$ for the none square-integrable function. This matches the observation that the central limit theorem predictions for the distribution only apply to square-integrable functions.

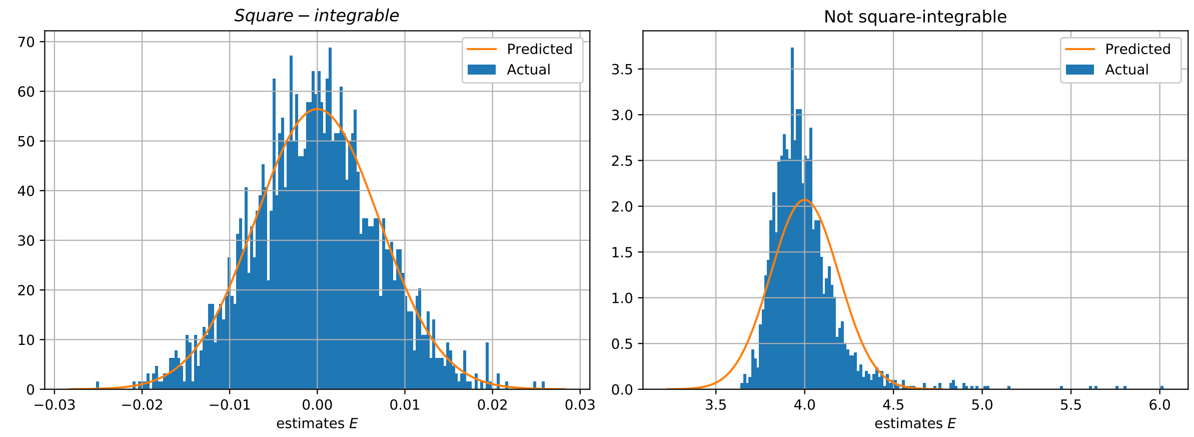

Figure 2.1.: Normalized histograms of 2000 runs of ordinary Monte Carlo integration with $N = 5000$ function evaluations each run, for the square-integrable (left) and and none square-integrable function (right) in eq. (2.5), and normal distributions with predicted variances.

2.2. Scaling of the Standard Deviation

The variance of Monte Carlo estimates in eq. (2.2) presents a central feature of Monte Carlo integration. Independently of the dimensionality, the standard deviation of the integration estimate scales like the inverse square root of function evaluations N, $\sigma_E = O(1/\sqrt{N})$. While the independence of the dimensionality is the strength of Monte Carlo integration, convergence with the inverse square root of N is slow compared to other integration methods. Using information about the integrand, there are variance reducing techniques that decrease the variance by a constant prefactor. Adaptive techniques are designed to learn about the integrand and decrease the prefactor during the integration process by adapting internal parameters.

Variance reducing techniques include stratified sampling, control and antithetic variates [3], and importance sampling (section 2.3). Common adaptive Monte Carlo techniques are the VEGAS-algorithm [11] and Multi-Channel Monte Carlo (section 2.4).

2.3. Importance Sampling

Importance sampling replaces the uniform distribution in plain Monte Carlo with a custom, fixed distribution, also referred to as *proposal* distribution or mapping (especially in the context of sampling methods). This can significantly improve the integration if the function is strongly peaked and information about the approximate location and shape of the peaks are known. The importance sampling method is mathematically based on a transformation of coordinates:

$$\int_V \mathrm{d}x\, f(x) = \int_V \frac{f(x)}{p(x)} p(x)\, \mathrm{d}x = \int_V \frac{f(x)}{p(x)}\, \mathrm{d}P(x). \tag{2.6}$$

The function $p(x)$ can be interpreted as probability distribution if $p(x) \geq 0$ and $\int_V p(x) = 1$. Given a method of selecting points x_n according to this distribution, the importance sampling Monte Carlo estimate becomes

$$E = \frac{1}{N} \sum_{n=1}^{N} \frac{f(x_n)}{p(x_n)}. \tag{2.7}$$

The variance of this estimate is given by [3]

$$\sigma^2(E) = \frac{1}{N} \int \mathrm{d}x\, p(x) \left(\frac{f(x)}{p(x)} - I \right)^2 = \frac{\sigma^2(f/p)}{N}, \tag{2.8}$$

with the sample estimate for the integrand variance

$$S^2(f/p) = \frac{1}{N} \sum_{n=1}^{N} \left(\frac{f(x_n)}{p(x_n)} \right)^2 - E^2. \tag{2.9}$$

A good choice of mapping p closely resembles f, thus reducing the integrand variance. The optimal choice would be $p(x) \propto f(x)$, for which the variance becomes zero (in this case the problem would already be analytically solved, however).

2.3.1. The Camel Distribution

The *Camel* distribution [11], consisting of two spatially separated Gaussian peaks, will serve as a toy problem for a target distribution (i.e. integrand) f. The probability density is defined as the superposition of two Gaussian distributions

$$f_{\text{camel}}(x) = \frac{1}{2}\left(\exp\left(-\frac{1}{2}z^{\intercal}\Sigma_1^{-1}z\right)\Bigg|_{z=x-\mu_1} + \exp\left(-\frac{1}{2}z^{\intercal}\Sigma_2^{-1}z\right)\Bigg|_{z=x-\mu_2}\right) \tag{2.10}$$

with the default values

$$\Sigma_1 = \Sigma_2 = \sigma^2\mathbb{1} = \frac{0.1^2}{2}\mathbb{1}; \qquad \mu_1 = \left(\frac{1}{3}, \frac{1}{3}, \dots\right)^{\intercal}, \qquad \mu_2 = \left(\frac{2}{3}, \frac{2}{3}, \dots\right)^{\intercal} \tag{2.11}$$

for the covariance matrices and the means in any number of dimensions (here, the one and two dimensional versions will be used).

The HMC method introduced in section 4.1 requires the gradient of the potential defined as $V(x) = -\log f(x)$. For the Camel distribution the potential gradient is

$$\nabla V_{\text{camel}}(x) = -\frac{\nabla f_{\text{camel}}(x)}{f_{\text{camel}}(x)}, \tag{2.12}$$

where the gradient of the Camel distribution is given by

$$-2\nabla f_{\text{camel}}(x) = \Sigma_1^{-1}z \cdot \exp\left(-\frac{1}{2}z^{\intercal}\Sigma_1^{-1}z\right)\Bigg|_{z=x-\mu_1} + \Sigma_2^{-1}z \cdot \exp\left(-\frac{1}{2}z^{\intercal}\Sigma_2^{-1}z\right)\Bigg|_{z=x-\mu_2}. \tag{2.13}$$

2.3.2. Effects of Different Probability Distributions

For analyzing the behavior of sampling and integration methods discussed here, the continuous, square-integrable camel distribution f_{camel} of eq. (2.10) with the parameters as in eq. (2.11) is considered (in one dimension).

Fig. 2.2 shows the deviation of importance sampling integration estimates from the exact value over a range of sample sizes on a log-log plot, using two different probability distributions, and plain Monte Carlo (which is equivalent to a uniform distribution over the whole volume). The root-mean-square deviations are obtained from 50 executions of the integration methods for each number of function evaluations. In the log-log plot, a scaling of $\sigma_E = c \cdot N^m$ for the standard deviations would show as a straight line with slope m, shifted up by c. According to eq. (2.8) the power is expected to be $m = -1/2$.

The first distribution p_{imp} is a Camel distribution with parameters slightly altered from the default parameters (here denoted as parameters without superscript; using the exact default parameters would make the distribution optimal):

$$\mu_1^{\text{imp}} = 0.9\mu_1, \qquad \mu_2^{\text{imp}} = \mu_2; \qquad \Sigma_1^{\text{imp}} = \Sigma_1, \qquad \Sigma_2^{\text{imp}} = \sqrt{1.1}\Sigma_2. \tag{2.14}$$

The first peak of this distribution is translated, the second peak is enlarged by 10 % relative to the integrand. The second distribution p_{var} is, up to a normalization factor, chosen in correspondence with the expression for the variance of a function, eq. (2.3):

$$p_{\text{var}}(x) \propto \big(f(x) - 1\big)^2 , \tag{2.15}$$

with $f = f_{\text{camel}}$ being the integrand (a Camel distribution with standard parameter values).

The plot shows the root-mean-square deviations of the estimates from the exact value, obtained from 50 executions of the method for each number of function evaluations. The predicted standard deviation shown is the square root of the mean of predicted variances, over the same number of executions.

This example illustrates the importance of minimizing the variance of f/p in choosing p. The distribution p_{var} emphasizes regions of f with high variance and does not closely resemble f, which leads to even worse results than a uniform distribution (plain Monte Carlo). The imperfect mapping p_{imp} proves best, as it most closely resembles the integrand. The plot also demonstrates that importance sampling only decreases the variance by a factor, but does not change the overall $N^{-1/2}$ scaling (the slope is approximately equal for all distributions), as noted earlier.

The predicted and actual deviations for the p_{cov} distribution are scattered significantly because the effective integrand f/p_{cov} goes to infinity close to the roots of p_{cov}, where f is nonzero. For further analysis it may be insightful to study the distribution of (predicted) variances for Monte Carlo techniques, since the variance estimates should not be subject to as broad a distribution as the estimates themselves. If they were, the predicted variance for a single execution could not be taken as a reliable measure of the estimate's quality.

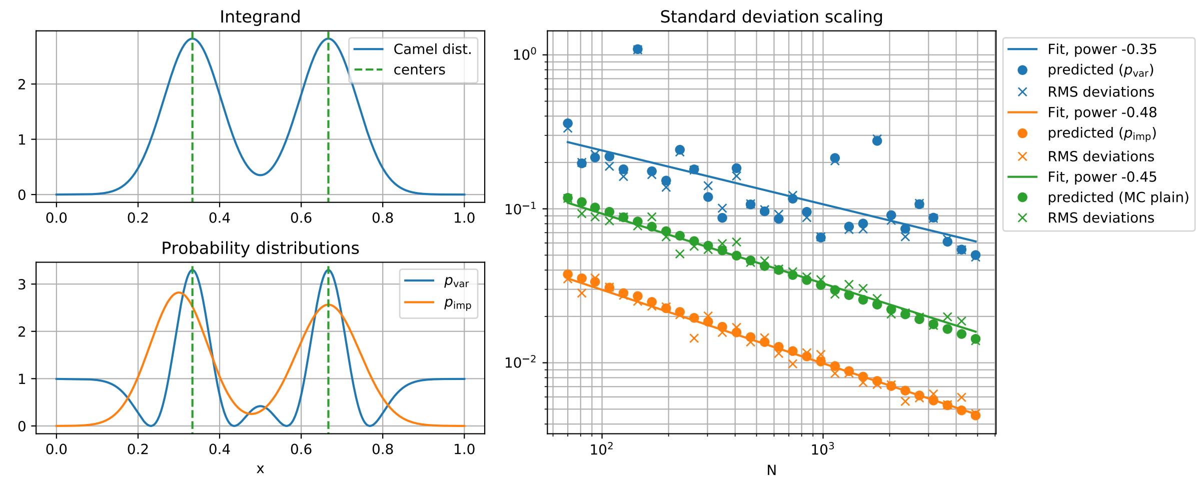

Figure 2.2.: Log-log plot (right) of the RMS deviation over 50 executions of importance sampling Monte Carlo integration of the camel distribution (top left, eq. (2.10)), using different probability distributions (bottom left, eq. (2.14), (2.15)). The linear (in log-log coordinates) fit was done on the actual RMS deviations.

2.3.3. Generating Samples for Monte Carlo Integration

Importance sampling requires a method to generate a sample according to a probability distribution p. While there are several algorithms (see section 3) for generating samples according to an arbitrary distribution (which will later be employed to generated unweighted events according to f), these methods are generally expensive. For the application relevant here, the distribution(s) for importance (and multi-channel) Monte Carlo p will be chosen according to limited information about the integrand such as approximate positions and sizes of peaks. It is therefore reasonable to limit the choice of p to distributions for which there is a known inverse and the inverse transform method is applicable, or for which special sampling methods exist.

The inverse transform method. In one dimension the probability distribution function $p(x)$ has a corresponding cumulative distribution function $P(x)$ mapping onto $[0, 1]$. If the inverse of P is known, samples can be generated by using common pseudo-random number generators to choose values $u \in [0, 1]$ and then compute $x = P^{-1}(u)$, which will have the desired distribution.

2.4. Multi-Channel Importance Sampling Monte Carlo

Multi-channel Monte Carlo is an adaptive Monte Carlo integration technique based on importance sampling. Instead of one probability distribution, the multi-channel technique comprises a set of distributions p_k (called channels) with respective weights α_k. For each step, a channel is chosen randomly with probability α_k, and x_i is (pseudo-) randomly selected according to the chosen channel. The integration process is split into iterations with N_j function evaluations in each, such that after one iteration the channel weights can be updated to minimize the variance of f/p, where $p(x) = \sum_{k=1}^{m} \alpha_k p_k(x)$ is the overall or total probability distribution. For each iteration define

$$W_{j,k}(\alpha) = \int_V \mathrm{d}x \; p_k(x) \left(\frac{f(x)}{p(x)}\right)^2 \approx \frac{V}{N_j} \sum_{i=1}^{N_j} \left(\frac{f(x_i)}{p(x_i}\right), \tag{2.16}$$

$$W_j(\alpha) = \int_V \mathrm{d}x \; p(x) \left(\frac{f(x)}{p(x)}\right)^2 \approx \sum_{k=1}^{m} \alpha_k W_{j,k}, \tag{2.17}$$

where the right sides are the Monte Carlo estimates. $W_{j,k}$ can be understood as the contribution of channel j to the total variance of the integrand. To reduce the variance, channels that contribute disproportionately much should have a larger weight. The channels are updated according to [12][3]

$$\alpha_k^{new} = \frac{\alpha_k (W_{j,k}(\alpha))^\beta}{\sum_k \alpha_k (W_{j,k}(\alpha))^\beta}, \tag{2.18}$$

with β ranging from 1/2 to 1/4. Analogous to importance sampling the integration estimate for one iteration is [3]

$$E_j = \frac{V}{N_j} \sum_{i=1}^{N_j} \frac{f(x_i)}{p(x_i)}, \tag{2.19}$$

$$\sigma^2_{E_j} = \frac{W_j(\alpha) - I^2}{N}. \tag{2.20}$$

Since the estimate is independent of the channel weights, the estimates from all iterations can be combined into a total estimate:

$$E = \frac{1}{N} \sum_j N_j E_j, \tag{2.21}$$

$$\sigma^2(E) = \frac{\sum_j \frac{Nj}{N} W_j - I^2}{N}, \tag{2.22}$$

where $N = \sum_j N_j$. Depending on the problem it might be useful to have an initial phase of weight optimization that does not contribute to the total estimate, or an additional final phase in which the weights are not updated.

2.4.1. Channel Weight Optimization Process

The multi-channel optimization algorithm is most useful, if the relative heights or positions of peaked areas of the integrand are not or only partially known.

Returning again to the Camel distribution from section 2.3.1 the case of unknown positions of the peaks can be explored. Assuming the exact locations of the peaks are unknown, and the peak structure is known to be Gaussian, the problem can be approached by selecting multiple Gaussian distributions as channels, with randomly chosen centers in the vicinity of the peaks. By adapting the channel weights, the integration method will increase the weight of channels close to the peaks and decrease weights of channels that barely contribute.

Fig. 2.3 shows the deviations of multi-channel integration estimates using as channels 15 Gaussian distributions with centers randomly chosen over the integration space, where each data point is obtained over 20 executions. The algorithm is set to do 1000 evaluations in each iteration and update the weights in each. For comparison, the standard deviations of a plain Monte Carlo integration are shown.

The standard deviations for the multi-channel variant is larger than for the plain Monte Carlo integration, for small number of integration steps N. This corresponds to the fact that the initial combined probability distribution, as visible from the right plot, approximates the integrand even worse than a uniform distribution (the initial distribution is close to minimal for the left peak). For total sample sizes approximately $10^3 < N < 10^4$ the convergence of multi-channel Monte Carlo is faster than $O(\sqrt{1/N})$, but levels off for $N > 10^4$ and eventually assumes a

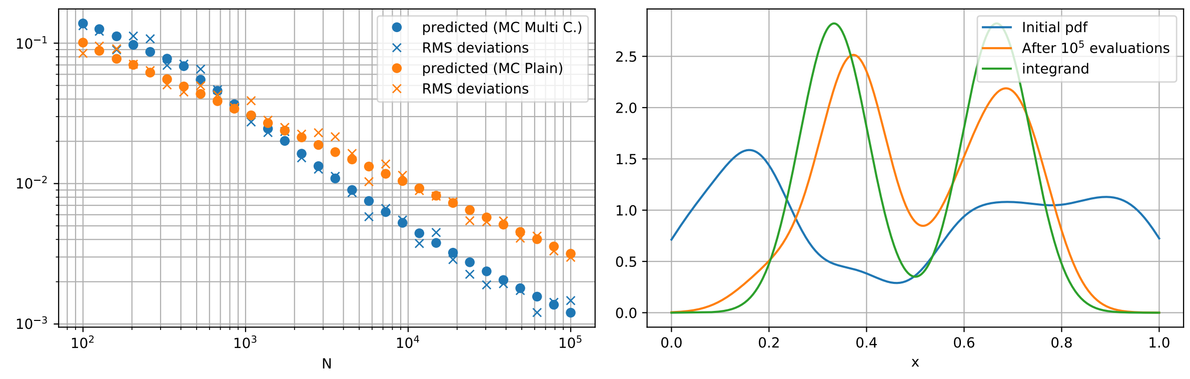

Figure 2.3.: Log-log plot (left) of deviations for multi-channel (using $\beta = 0.5$) and plain Monte Carlo integration of the one-dimensional Camel distribution in eq. (2.10) with parameters as in eq. (2.11). Each iteration comprises 1000 samples, each data point is the mean over 20 executions. On the right are the initial and final overall probability distributions. The channels are Gaussian distributions with variance $\sigma^2 = 0.005$ (the same as one of the peaks in the Camel distribution) and centers randomly chosen within $[0, 1]$.

$O(\sqrt{1/N})$ behavior. This corresponds to the ideal channel weights being found and multi-channel sampling effectively becoming importance sampling, which converges by a constant factor faster than plain Monte Carlo.

3. Generating Samples According to Arbitrary Distributions

The cheapest and thus preferred method for sampling according to non-uniform distributions is the inverse transform method. It is, however, only applicable if the inverse of the distribution is known, which is not generally the case. Since the goal is to generate unweighted samples according to a function f related to the matrix element about which (generally) little is known, other methods must be used.

In the following two sampling methods are introduced that generate unweighted events: the acceptance-rejection method and Markov chains (specifically the Metropolis Hasting algorithm). Neither method requires the probability distribution function they sample to be normalized. The MC^3 algorithm in section 3.4 provides a framework to take advantage of a local Markov chain while reducing correlation using the multi-channel configuration (from section 2.4) in analogy to the proposal in the acceptance-rejection method.

3.1. Acceptance-Rejection Sampling

Acceptance-Rejection sampling generates unweighted samples according to an (unnormalized) distribution $f(x)$ using a distribution $p(x)$ for which a sampling method is known. A constant C must be known such that $f(x) \leq Cp(x)$ everywhere.

Samples generated according to $p(x)$ would have an inherent weight of $f(x)/p(x)$. They can be unweighted by only accepting a chosen sample value x with a probability $f(x)/(Cp(x))$. If a sample is rejected, a new x is chosen according to $p(x)$. These two steps are repeated until a value is accepted. Samples generated this way will follow the desired distribution $f(x)$ [3].

The biggest drawback of this method is that it requires detailed knowledge about the desired distribution to be efficient. If the sampling distribution does not closely resemble the desired distribution, many sampled values will be rejected and thrown away. The constant C must be as small as possible for the method to be efficient, which requires knowledge of the maximal value of $f(x)/p(x)$. While the acceptance-rejection algorithm itself will not be further considered, as it lacks the extensibility of Markov chains described in the following section, it is analogous to the unweighting of proposals in the Metropolis importance sampling update of the MC^3 algorithm in section 3.4.

3.2. Markov Chains

A Markov Chain is a sequence of elements $x_1, x_2, ...$ where the distribution of x_{k+1} only depends on the previous element x_k. This conditional distribution $P(x_{k+1}|x_k)$ is called transition probability distribution. The distribution of x_1 is called initial distribution. A distribution h that is preserved under the conditional distribution of the Markov chain, $P(x|y)h(y) = h(x)$, is called its stationary distribution.

A Markov Chain is said to be stationary if the marginal distribution of x_n does not depend on n. It is time-homogeneous if $P(x_{k+1}|x_k)$ does not depend on k. A time-homogeneous Markov Chain has a unique stationary distribution [1]. If the Markov Chain is ergodic (every state can be reached within a finite number of steps) and the transition probability satisfies detailed balance with respect to the stationary distribution f,

$$f(x_1)P(x_2|x_1) = f(x_2)P(x_1|x_2), \tag{3.1}$$

the Markov Chain converges to the stationary distribution (equilibrium distribution) for any initial distribution [3][4].

To generate a sample according to a given distribution, a Markov Chain with this distribution as equilibrium distribution has to be constructed. To assure the Markov Chain converges, it is easiest to show detailed balance and use time-homogeneous transitions. A common method is the Metropolis-Hasting algorithm, of which the Metropolis algorithm is a special case.

3.2.1. Metropolis-Hasting Algorithm

The Metropolis-Hasting algorithm defines an update mechanism for generating a Markov Chain with a given desired (target) equilibrium distribution f. It requires a sampling method that proposes a new state x_{k+1} with a known conditional probability (proposal distribution) $q(x_{k+1}|x_k)$, and the initial value of the Markov chain.

To generate the next value x_{k+1} in the Markov Chain given x_k, a candidate y is chosen from the proposal distribution. The candidate is accepted with the acceptance probability

$$\alpha(y|x_k) = \min\left(1, \frac{f(y)q(x_k|y)}{f(x_k)q(y|x_k)}\right), \tag{3.2}$$

otherwise the next value is $x_{k+1} = x_k$. The chain generated this way can be shown to be ergodic and fulfill detailed balance, and thus converges to the desired equilibrium distribution f [4].

Metropolis Update. If the proposal distribution is symmetric, $q(x|y) = q(y|x)$, the acceptance probability simplifies to

$$\min\left(1, \frac{p(y)}{p(x_k)}\right). \tag{3.3}$$

This special case is called Metropolis update.

In the following the term *local* Metropolis (Hasting) update is used to describe updates for which the proposal distribution $q(x|y)$ is localized (explicitly dependent on the previous state y), as opposed to a *global* Metropolis Hasting update for which $q(x|y) = q(x)$. The local Metropolis update will often have a fixed proposal distribution of the form $q(x|y) = p(x - y)$. Similar to the acceptance-rejection method, sample points generated using a global Metropolis Hasting update are independent, but state repetitions are introduced when a proposal is rejected.

3.3. Analyzing Generated Samples

It is important to understand the behavior and the quality of generated samples. A basic understanding of the sample can be provided by the sample mean $\bar{x}$ and the sample variance S_x^2, which, in the limit, should converge to the mean μ_f and variance σ_f^2 of the distribution given by f (note μ_f and σ_f^2 do not refer to the mean and variance of the function f itself; they are values in x-space). A normalized histogram can show if the sample roughly follows the distribution. However, especially for more complicated distributions, these methods are unsatisfactory for a reasonable judgment of sample quality.

3.3.1. Lag-Autocorrelation

If the conditional proposal distribution used in the Metropolis-Hasting algorithm is local, the subsequent elements of the Markov Chain tend to stay in the vicinity of each other; the sequence performs a kind of random walk. The lag auto-correlation measures how similar (i.e. close) points are depending on their distance in the chain. For a high-quality sample, the lag-autocorrelation must be minimized (this can be achieved by e.g. mixing Markov Chains or by subsampling [4]).

The lag-k autocovariance is defined as

$$\gamma_k = \operatorname{cov}(x_i, x_{i+k}), \tag{3.4}$$

with the natural estimator

$$\hat{\gamma}_k = \frac{1}{N} \sum_{i=1}^{N-k} (x_i - \bar{x})(x_{i+k} - \bar{x}). \tag{3.5}$$

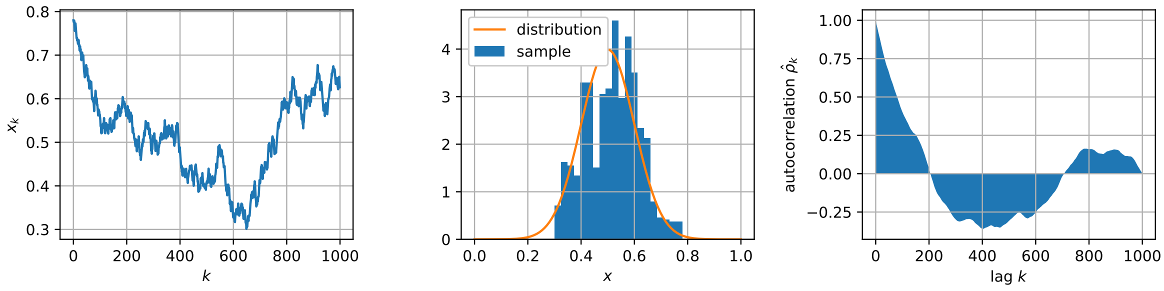

Figure 3.1.: Time series plot (left) of a Markov chain of length 1 000, sampled using a local Metropolis update with a Gaussian distribution ($\sigma = 0.1$, $\mu = 0.5$) as equilibrium distribution. A normalized histogram together with the equilibrium distribution of the sample is shown in the middle, with the autocorrelation function of the same run shown on the right. The candidate in the Metropolis update is chosen uniformly within an interval of length 0.015 around the previous value.

For any distribution property ξ the sample estimator is referred to as $\hat{\xi}$, if the distinction is made explicitly. The lag-k autocorrelation is the autocovariance divided by the variance:

$$\rho_k = \frac{\gamma_k}{\gamma_0}, \qquad \hat{\rho}_k = \frac{\hat{\gamma}_k}{\hat{\gamma}_0}. \tag{3.6}$$

Fig. 3.1 illustrates how a lag-autocorrelation plot can reveal poor sampling performance. A local Metropolis update is used to generate a sample according to a Gaussian target distribution. The proposal distribution is uniform on an interval of length 0.015 around the previous value, leading to a local selection of points clearly visible in the random walk-like time series plot. The lag-autocorrelation plot confirms the correlation of neighboring points (in higher dimensions and for more complicated target distributions, it is neither feasible nor insightful to plot the full Markov chain as done here). The lag-autocorrelation is used for the effective sample size in section 3.3.3, which summarizes the correlation of a sample into a single measure.

3.3.2. Bin-Wise Chi Squared Test

The bin-wise χ^2 statistic provides a quantitative measure for how well a given sample matches an arbitrary target distribution. For M bins and N_i of the samples falling into bin i with a theoretically predicted number n_i of points, the χ^2 value is [13]

$$\chi^2 = \sum_i^M \frac{(N_i - n_i)^2}{n_i}. \tag{3.7}$$

Assuming the number of points in each bin follows a normal (Poisson for large enough n_i; $\sigma^2 = n_i$) distribution, the above statistic is expected to follow a χ^2 distribution with $M - 1$ degrees of freedom (given the number of points for $M - 1$ bins, the number of points in the last bin is fixed by the total sample size). If the sample follows the target distribution, the number of bins in each bin follows a Poisson distribution. The assumption of normal distributed counts (required for the statistic to follow a χ^2 distribution) is only valid for counts large enough such that the Poisson distribution is close to normal. Practically, given a specific binning, all bins containing fewer than 10 observed or predicted sample points will be ignored (M then refers to the number of actually contributing bins).

Since the χ^2 distribution for any given degrees of freedom is analytically known, the corresponding p-value, denoted as $p(\chi^2)$ can be given. The p-value is the significance of obtaining a certain χ^2 value assuming as null hypothesis the sample follows the target distribution. It therefore gives the significance under which the target distribution can be rejected as describing the sample, and can be used as measuring the quality of a sample (larger values correspond to higher quality).

Especially for samples generated with a form of Metropolis algorithm one should be aware of the impact of different binning choices. The Metropolis algorithm has a non-zero chance of repeating the exact same value, which will for small bin-sizes lead to larger values of the statistic. As long as the chosen binning is fixed, the χ^2 statistic can be used to compare the quality of different samples of the same size, where smaller values of χ^2/dof imply closer resemblance to the equilibrium distribution (more accurately, smaller values imply larger statistical compatibility). An estimator for a reasonable binning choice used for the analysis here can be found in section A in the appendix.

For the sample distribution in fig. 3.1 it is $\chi^2/(M-1) = 6.97$ and $p(\chi^2) = 1.08 \times 10^{-13}$ (using 16 bins out of which $M = 14$ are valid; the plot shows 20 bins), which confirms poor quality due to the localized sampling.

3.3.3. Effective Sample Size

Given $x_{i=1,\dots,N}$ (generally correlated) Markov chain sample points, the effective sample size (ESS) measures how many independent sample points would be needed to estimate the mean μ_f of the target distribution f with the same variance as the Markov chain estimate using the x_i [6]:

$$ESS_f(x_i) = N \frac{\sigma^2\left[\frac{1}{N}\sum_{i=1}^{N} f(x_i)\right]}{\frac{1}{N}\,\sigma^2(f)} \tag{3.8}$$

This can be used to describe the performance of a Markov chain-based sampler, as it gives the number of independent sample points a correlated sample is effectively worth. Values near the total sample size N indicate close to independent sample points, while smaller values hint at considerable random-walk behavior. As in [6], the ESS is measured using the lag-autocorrelation ρ_k of eq. (3.6) using as cutoff N^{cutoff} the first time $\hat{\rho}_k$ drops under a value of 0.05:

$$E\hat{S}S_f(x_i) = \frac{N}{1 + 2\sum_{i=1}^{N^{\text{cutoff}}}(1-\frac{i}{N})\hat{\rho}_i}; \qquad N^{\text{cutoff}} = \max\{k = 1,\dots,N \,|\, \hat{\rho} < 0.05\}. \tag{3.9}$$

The cutoff is needed to prevent large values of the lag k for which the estimator $\hat{\rho}_k$ inevitably becomes noisy. To avoid underestimating the lag-autocorrelation, in its computation the sample mean $\bar{x}$ is replaced by either the known mean μ_f or an independent and more reliable estimate $\hat{\mu}_f$ [6]. For multiple dimensions the calculation of the ESS can be done independently for each dimension. To obtain a single value for the ESS of a given sample, the minimal (worst) value is chosen.

For the sample in fig. 3.1 it is $E\hat{S}S = 5.7$, which (compared to $N = 5\,000$ sample points) highlights the autocorrelation due to the random walk even more clearly than the bin-wise χ^2 or the lag-autocorrelation plot.

3.4. Multi-Channel Markov Chain Monte Carlo

The multi-channel Markov chain Monte Carlo (MC^3) method introduced in [1] combines the classical global Metropolis-Hasting sampler using the multi-channel Monte Carlo integration with a local Metropolis-Hasting algorithm. The basic goal is to increase the acceptance rate and thus performance of the sampling method, while maintaining relatively low lag-autocorrelation.

In the first phase, the function f is integrated over the sample space using multi-channel Monte Carlo. The channels used for the integration must be passed to the algorithm (and constructed using knowledge about the function such as sizes and locations of peaks, which physically correspond to resonances in the matrix element). As a result of the multi-channel integration the channel weights are optimized, leading to a combined probability distribution p_{IS} which approximates f. The total integral of f may be of interest to users of the algorithm, but is unimportant for the sample generation.

The sampling, which follows the integration phase, generates a Markov Chain by mixing two Metropolis-Hasting update mechanisms (making MC^3 a random scan Metropolis algorithm [4]). The first update mechanism is chosen with probability β, and is a global Metropolis-Hasting update with candidates proposed according to the distribution p_{IS} from the multi-channel integration (thus independently of the previous state in the chain). The second, chosen with probability $1-\beta$, is a Metropolis update that uses some local proposal distribution $p_{\text{loc}}(x|y)$. In the original paper suggests using a local proposal distribution, such as a symmetric Gaussian distribution, the width of which may be adapted to reach a certain target acceptance rate (section 3.4.2). The combined update mechanism is reversible and has the desired equilibrium distribution f [1].

3.4.1. Parameter Choice

The main parameter of the MC^3 method is β, which sets the probability of using the importance sampling update in the Markov chain. Since the candidates in this update mechanism follow the optimized distribution p_{IS} of the integration channels, larger values of β correspond to higher confidence in this distribution being a good approximation of f.

Further properties of the MC^3 method include the number of iterations for which to integrate the function (and thus optimize the channels), and the size and shape of the local proposal distribution in the local update mechanism. These properties are, however, not generally referred to as parameters as they may be adapted and chosen by the algorithm. The original paper proposes to change the size of the local proposal distribution until an acceptance rate in the range $[0.25, 0.5]$ is reached (see section 3.4.2) [1].

As an example equilibrium distribution the modulated sine-squared target distribution

$$f(x) = \sin^2(2\pi x) \cdot \sin^2(10 \cdot 2\pi x) \tag{3.10}$$

is considered. Fig. 3.2 shows the impact of β on the sampling of this distribution. The proposal distribution in the second update method is fixed to a uniform distribution within an interval of length 0.01 around the previous chain element. The distributions from fig. 2.2 were used for the channels. The first distribution shown is of a sample generated with $\beta = 0.6$, which slightly favors importance sampling. It can be qualitatively seen that the distribution matches the desired equilibrium distribution reasonably well. The second sample with $\beta = 0.01$ strongly favors the local Markov chain update. As can be seen in the time series plot, the Markov chain performs a random walk in a small area and tends to stay within one of the peaks, interrupted by few jumps due to the global update. This leads to some, especially small peaks, to be ignored and others receiving a too large weight. The χ^2/dof values for the two distributions in fig. 3.2 are 6.895 ($\beta = 0.6$) and 93.302 ($\beta = 0.01$), confirming the difference in sampling quality.

3.4.2. Adaptive Selection of the Proposal Distribution Width

If the MC^3 method is used with a fixed local proposal distribution (e.g. Gaussian), the width of this distribution must be chosen. The acceptance rate should be in the range of $[0.25, 0.5]$ in order to obtain reasonable performance of the local update [1], which can be used as a measure for adequate proposal widths.

The stochastic optimization method with vanishing adaptation algorithm introduced in [14] can be used to iteratively update the covariance of the proposal distribution in a way that a given measure (here the acceptance rate) converges to a desired value α. Given the measure r_t for a certain iteration step t, define

$$H_t = \alpha - r_t. \tag{3.11}$$

If the parameter l that is to be optimized is updated according to

$$l_{t+1} = l_t - cH_t t^{-\kappa}, \tag{3.12}$$

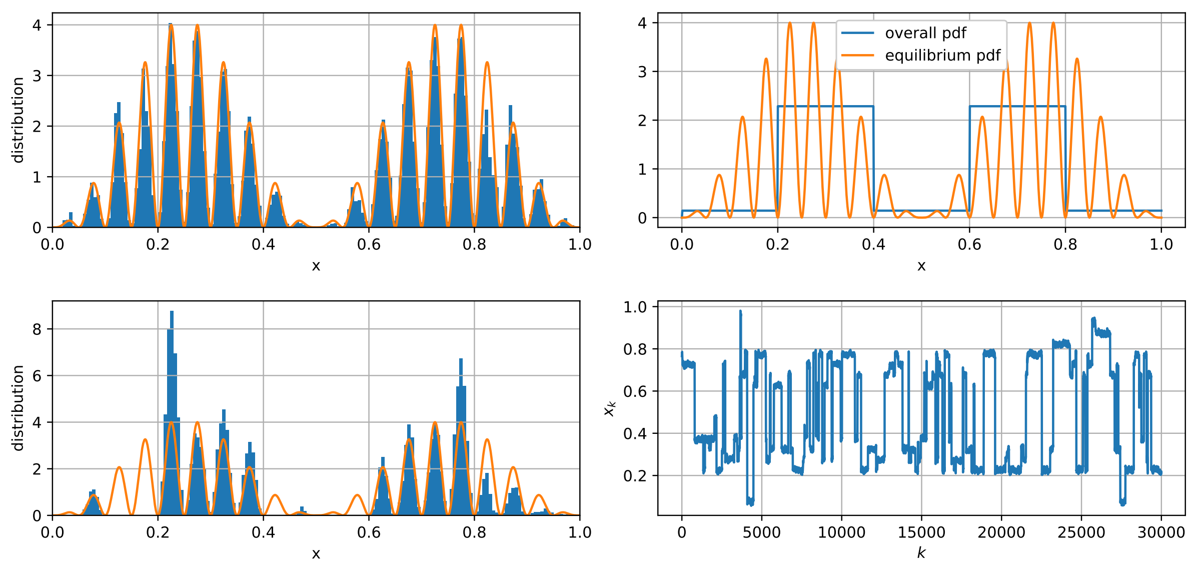

Figure 3.2.: MC^3 sampling results using an imperfect step function approximation as p_{IS}, uniform local sampling within a range 0.01 around the previous state, and the equilibrium distribution of eq. (3.10). The proposal distribution p_{IS} as well as the normalized equilibrium distribution are shown on the top right. On the left are the normalized histograms of 30 000 sample points for $\beta = 0.6$ (top) and $\beta = 0.01$ (bottom) together with the equilibrium distribution. The bottom right shows the time series plot of the Markov Chain generated for $\beta = 0.01$.

for some proportionality c and parameter $\kappa \in (0.5, 1]$, r_t will converge to α [6]. To make the resulting Metropolis sampler stationary, the actual sampling may be preceded by a *burn-in phase*, in which the adaptation is done and all generated samples are discarded. For the actual sample generation the proposal distribution would be fixed.

Fig. 3.3 shows the acceptance rate and width of a Gaussian proposal distribution in the adaptation process (corresponding to a burn-in phase) for a Camel target distribution (eq. (2.10) and (2.11)). The adaptation for the larger value of $\kappa = 0.9$ is significantly faster, and it is visible the burn-in phase could be terminated after 2000 steps as the acceptance rate no longer changes (within the acceptable range). The final value of the width for the $\kappa = 0.9$ run is

$$\sigma_{\text{local}} = 0.504. \tag{3.13}$$

3.4.3. Sampling the Camel Distribution

The Camel distribution introduced in section 2.3.1 will serve as a target distribution to evaluate and compare sampling efficiency of the various methods. To know if using more advanced Hamiltonian methods (section 4.1) within the MC^3 framework leads to improvements compared to the basic MC^3 version, first the basic version must be analyzed. The Camel distribution is chosen as example because it is known analytically (including its gradient) and for any number of dimensions, while being complex enough to demonstrate properties of the different

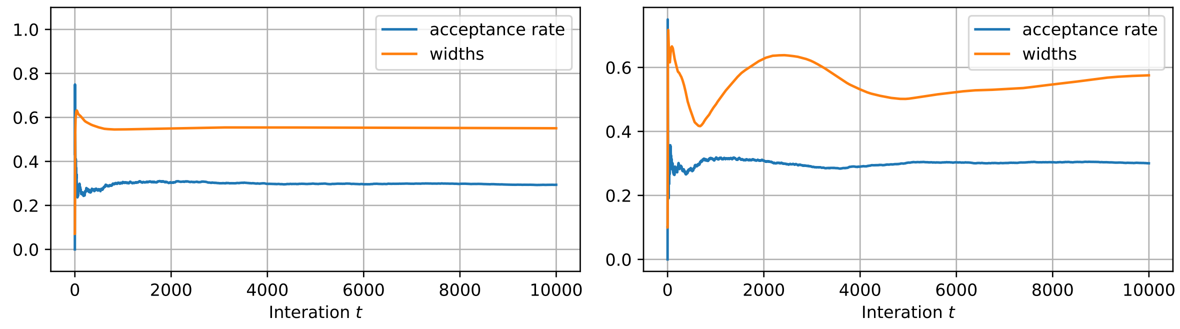

Figure 3.3.: Adaptation process according to eq. (3.12) for the width of a local Gaussian proposal distribution for a Camel as target distribution. On the left it is $\kappa = 0.9$, on the right $\kappa = 0.51$. The target acceptance rate is 0.3, and the proportionality factor $c = 0.5$ was used.

sampling methods. The two peaks of the Camel distribution are spatially separated which makes the distribution difficult for local Metropolis samplers, and (using the default parameters) sufficiently narrow to make methods such as acceptance-rejection (section 3.1) inefficient.

Using a two-dimensional Camel distribution as target, the variance of a local Gaussian proposal distribution, in analogy to the analysis in section 3.4.2, for an acceptance rate of 0.3, can be found to be approximately

$$\sigma^2_{\text{local}} = 0.0208. \tag{3.14}$$

Generally, the width of the Gaussian distribution used as proposal here is described using a covariance matrix. Since the algorithm presented in eq. (3.12) is limited to adapting the overall size of the covariance matrix, it is here simply represented as the scalar variance (the corresponding covariance matrix is $\sigma^2\mathbb{1}$). The results for the basic MC^3 algorithm, together with other sampling methods, can be seen in fig. 4.4.

4. Informed Sampling Methods

4.1. Hamiltonian Monte Carlo

The performance of a Metropolis sampler is maximized if subsequent states are as uncorrelated, and the acceptance rate is as high as possible. For a Metropolis update with a fixed local proposal distribution, reducing the proposal width increases the acceptance rate but also increases the random-walk behavior and thus the lag-autocorrelation. Hamiltonian Monte Carlo (HMC), originally introduced as hybrid Monte Carlo [8], uses local information about the distribution to decrease the random-walk behavior while significantly increasing the acceptance rate.

In analogy to statistical mechanics the potential to the probability distribution function f is defined as $V(x) = -\log f(x)$ (note that classical mechanics is invariant under shifting the potential by an additive constant, and the Metropolis algorithm is only sensitive to probability ratios, thus f does not have to be normalized). Using a Hamiltonian of the form

$$H(x, \pi) = V(x) + K(\pi), \tag{4.1}$$

an artificial momentum variable π (of the same dimension as x) is introduced with K such that the distribution $p(\pi) = e^{-K(\pi)}$ is is Gaussian [5]:

$$K(\pi) = \frac{1}{2}\pi^T M^{-1} \pi. \tag{4.2}$$

The covariance matrix M corresponds to a mass and can be any symmetric, positive-definite matrix (in the following applications it is always taken to be a scalar multiple of the identity and *mass* refers to a scalar value). The dynamics of the system is given by Hamilton's equations

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = \frac{\partial H}{\partial \pi_i} = \frac{\partial K(\pi)}{\partial \pi_i}, \qquad \frac{\mathrm{d}\pi_i}{\mathrm{d}t} = -\frac{\partial H}{\partial x_i} = \frac{\partial V(x)}{\partial x_i}. \tag{4.3}$$

In Hamiltonian dynamics, the value of the Hamiltonian $H(x, \pi)$ remains constant as the system evolves over time, preserving the total probability

$$p(x, \pi) = e^{-(-\log f(x))} e^{-K(\pi)} = e^{-H(x,\pi)}. \tag{4.4}$$

This would theoretically lead to a perfect acceptance rate, however numerical mistakes (and possibly an inexact potential gradient) reduce this number.

In practice, a discretization of the Hamiltonian equation must be chosen to simulate the dynamics, introducing the number of simulation steps and the step size as parameters (such that the simulation would correspond to a propagation by *steps* · *step size* in time). The Hamilton dynamics itself is volume preserving, such that the Metropolis acceptance probability does not have to take volume change into account. It is important to choose a simulation method preserving this property, in addition to ensuring time reversibility required for detailed balance of the Metropolis update. A common choice satisfying these conditions [5], and the one used for the implementation here, is a version of the leapfrog algorithm.

```
def kin(p, M):
  # potential corresponding to a Gaussian distribution, −log(pdf)
  return $p_i M^{-1}_{ik} p_k$ / 2  # sum over i and k

def kin_gradient(p, M):
  # gradient of kin
  return $M^{-1}_{ik} p_k$  # sum over k

def leapfrog(q, p, pot_gradient, M, steps, step_size):
  q, p = copy(q), copy(p)  # don't override input variables

  for i in range(steps):
    p −= step_size/2 * pot_gradient(q)
    q += step_size * kin_gradient(p_next, M)
    p −= step_size/2 * pot_gradient(q)
  return q, p

def next_state(q, pdf, pot_gradient, M, steps, step_size):
  # generate the next state in the Markov chain

  # Gibbs update of the momentum variable (sample known distribution)
  p = gaussian.rvs(q.size, cov=M)

  # Hamiltonian (Metropolis) update
  q_next, p_next = leapfrog(q, p, pot_gradient, M, steps, step_size)
  accept = pdf(q_next) * gaussian.pdf(p_next, M) / (pdf(q) * gaussian.pdf(p, M))
  if random() < accept:
    return q_next  # accept q_next with the probability 'accept'
  return q
```

Algorithm 4.1: Python pseudo-code for a Hamiltonian Metropolis update (`next_state`). The variables `pdf` and `pot_gradient` refer to $f(x)$ and $\partial V(x)/\partial x$ of the target distribution, the momentum π is referred to as `p`.

The Hamiltonian update together with the leapfrog method is described in Python pseudo-code in alg. 4.1. Sampled from its target distribution (Gibbs update), the momentum π together with the previous state x of the Markov chain is propagated using the leapfrog algorithm to obtain a proposal. The final x value is accepted with the Metropolis acceptance probability given by the ratio of the combined probability of x and π, compensating for inexact time evolution (due to numerical errors or an approximate gradient). The HMC update defined in this way fulfills detailed balance and is ergodic (for most parameter choices, see the discussion in section 4.1.2) [5]. In addition to the target distribution f, the algorithm requires, at least in numerical form, an expression for the gradient of the potential $V(x) = -\log f(x)$.

4.1.1. Challenges

Potential Gradient. Besides the target distribution f, the simulation of the Hamilton dynamics requires the gradient of the potential $V(x) = -\log f(x)$. For applications in high energy physics, the function f is only numerically available and the potential gradient cannot be analytically derived. A naive approach would be to obtain the gradient by evaluating f and computing the differences of the logarithm of these values. Assuming the (phase-) space of x is K dimensional, f needs to be evaluated at least K additional times (once for a step in the direction of each dimension). Each step in the leapfrog algorithm requires two evaluations of the gradient, thus making the minimal number of function calls needed for a sample of size N

$$N_f^{HMC} = N + 2N \cdot K \cdot \text{steps}. \tag{4.5}$$

The number of steps is often chosen larger than 10 and the dimensionality is generally high, making HMC (using this construction for the gradient) multiple orders of magnitude more expensive than a Metropolis update with fixed proposal distribution. Since the objective is to minimize calls to f, other methods such as the extreme learning machine introduced in section 4.3 should be considered to obtain an approximation of the potential gradient.

Parameter Space. Another complication introduced by HMC is the presence of two additional parameters, the step size and the number of steps for the leapfrog simulation (the mass is commonly fixed, see section 4.1.2). If used in combination with the MC^3 algorithm, three parameters need to be explored for an optimal/acceptable configuration (see sections 4.1.3 and 4.2.1). Especially for applications where Monte Carlo methods are used in the background, there has to be an automated algorithm to choose at least a subset of these parameters. The No-U-Turn sampler introduced in [6] chooses the number of simulation steps for each proposal dynamically, maximizing the distance of adjacent states by continuing the simulation as long as the motion moves away from the starting position. Additionally a method based on the dual averaging method of [15] is suggested to automatically tune the step size, eliminating the need to manually choose any parameters introduced by HMC.

Zero-Probability Areas. Using HMC in combination with a probability distribution with extended areas of zero probability can be problematic, since generally the potential would be infinite and the gradient not well defined. A special case of this is a boundary constraint, which can be resolved by reflecting at the borders or using a mapping such as Spherical HMC as introduced in [9].

The situation is slightly different if the gradient was learned computationally (section 4.3). If the target distribution is localized by peaks decreasing continuously to zero, the HMC dynamics will (for well-tuned parameters) never or rarely move into problematic areas. Boundary constrained targets do pose a challenge in the sense that the learned gradient may push the Hamiltonian dynamics into zero probability regions leading to low acceptance rates (section 5.2), and the solutions mentioned above should be applied.

4.1.2. Parameter Choice

For an overview of the impact of the various parameter choices introduced by HMC, in the following a two-dimensional Gaussian distribution with mean $\mu = (1/3, 1/3)^\intercal$ and variance $\sigma^2 = 0.005$ (i.e. with covariance matrix $\sigma^2 \mathbb{1}_{2\times 2}$) is considered as target distribution (this choice corresponds exactly to one of the peaks in the Camel distribution used in sections 4.1.3 and 4.2.1). The potential and potential gradient for the Gaussian distribution are analytically known (see `kin_gradient` in alg. 4.1). For this target distribution, the solutions of the Hamilton dynamics are ellipses. It is important to note that the Gaussian is a localized distribution that quickly and continuously approaches zero away from the center such that no border phenomena have to be considered.

Fig. 4.1 shows the measures introduced in 3.3 of samples generated using HMC for different masses and number of simulation steps choices over a range of step sizes. A periodic behavior, due to the symmetry of the target distribution (the HMC dynamics is elliptical), is observable. For small step sizes, the update only proposes states in the vicinity of each other leading to large χ^2, small $p(\chi^2)$ and small ESS values. By increasing the step sizes the ESS increases to its maximal value, until it drops again at a point where the ellipses of the motion close, making the Markov chain stay close to a given starting point. This behavior is further illustrated in fig. 4.2, which shows the trajectories arising from the HMC algorithm corresponding to configurations 7 and 8 in fig. 4.1. Both figures seem to suggest choosing a different mass corresponds to re-scaling the behavior over the step size range, as no new features are apparent. A closer inspection of the algorithm defined in alg. 4.1 indeed confirms that a linear scaling of the mass (covariance matrix) M by a factor C (leading to a scaling in the momentum variable by $\sqrt{C}$) is just compensated by scaling the step size by a factor $\sqrt{C}$. This behavior can also be numerically confirmed by comparing or re-scaling the plots in fig. 4.1.

Based on this observation, the mass of HMC can generally be fixed, reducing the parameters to the step size and step count (in the following sections it will be $M = \mathbb{1}$). If all degrees of freedom of the mass matrix are considered, this observation only applies to the overall scale and the relative scale of the different dimensions may still be optimized. A similar scaling behavior is observable for the step count with respect to the step size, but because the simulation method is inexact, and numerical errors occur, neither can be set to a fixed value without potentially loosing relevant ranges. While configurations 1 and 8 (or 5 and 10) in fig. 4.1 show the same range and scale of qualitative behavior for χ^2 and ESS (corresponding to analogous trajectories), the configuration with fewer number of steps show significant deviations of the acceptance rate from 100 %. This corresponds to simulation errors in the leapfrog simulation which increase for larger step sizes. Following the trend apparent from configuration 1, an exemplary HMC run with mass 0.1, 10 steps, and a step size of 0.06 for 1000 samples led to an acceptance rate of 0.0. The same (analytical) Hamiltonian dynamics for a total time $\textit{steps} \cdot \textit{step size} = 0.6$ is simulated in configuration 5 with a step size of 0.015 and 40, where the acceptance rate is close to 100 % and both the χ^2 and ESS measures indicate successful sampling.

While this artificial problem illustrates general features and relations of the HMC parameters, it is not possible to extract a general choice for the step size (range) or the number of steps (e.g. the periodicity observed in fig. 4.1 only emerges for distributions with periodic solutions to the associated Hamiltonian dynamics). The Camel distribution examined in the following section, however, is equivalent to two spatially separated Gaussian distributions with the same variance as the one considered here. Following from this discussion, it is therefore reasonable to limit the parameter space to one configuration; corresponding to configuration 7 in fig. 4.1, a mass of 1, 40 simulation steps, and the range $[0, 0.01]$ of step sizes will be used.

4.1.3. Sampling the Camel distribution

For a slightly more realistic example, and to understand the limitations of the HMC method, the Camel distribution introduced in section 2.3.1 is considered. Figure 4.3 shows the ESS, the χ^2 (and corresponding p-) values, and the acceptance rate of samples generated using HMC for a single Gaussian distribution and a Camel distribution (with the default parameters from section 2.3.1; the Gaussian corresponds to one of the peaks). The parameters of the HMC update are40 simulation steps and the range $[0, 0.01]$ of step sizes, in dependence of which the quality measures are shown (based on results from the previous section, this choice of parameters is sufficient to explore the sampling of the Gaussian distribution). For the two distributions a similar qualitative behavior is observable, as the areas of larger p-values overlap, and the high-χ^2 middle section (corresponding to periodic ellipses) are visible and align in both plots. However, the ESS as well as the χ^2 p-values are generally smaller for the Camel distribution compared to the single Gaussian distribution, which is due to the significant spatial separation of the two peaks that makes it unlikely for the Hamiltonian dynamics to move from one peak to the other. The plot for the Camel distribution is not symmetric like the one for a Gaussian distribution; the ESS increases for larger step sizes, beyond the point where it drops for a Gaussian distribution, and the p-values seem to be larger for larger step sizes. This corresponds to a larger likelihood of the HMC dynamics to cross over to the other peak.

By considering larger values for the step size, and carefully observing the increased numerical error, one may be able to further optimize the HMC algorithm to sample the Camel distribution. This would, however, lead off-topic as the Camel is only an artificial toy problem used here to

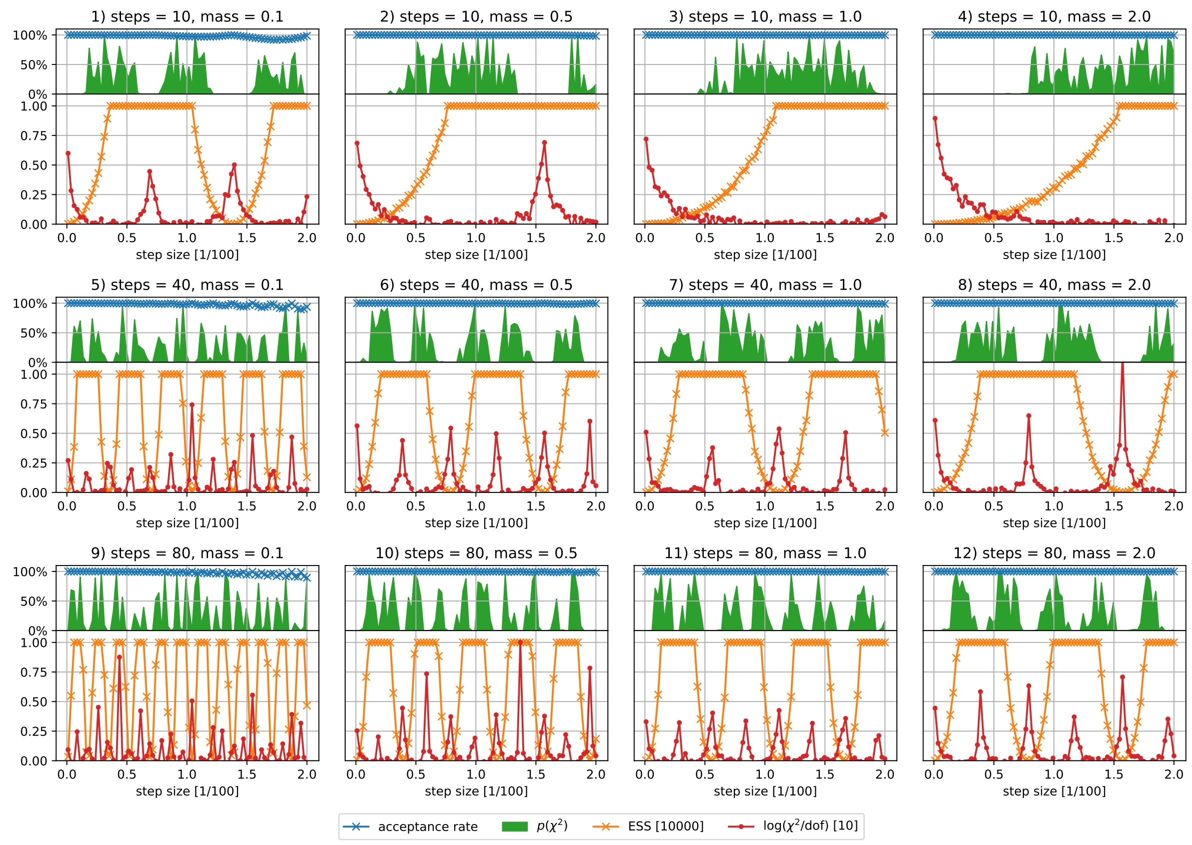

Figure 4.1.: Sampling results of HMC using various parameter choices for a Gaussian target distribution with $\mu = 1/3$ and $\sigma^2 = 0.005$. Each plot shows the acceptance rate of the HMC update, the ESS, the logarithm of the χ^2 value (on a linear scale large χ^2 values would obscure the qualitative behavior), and the corresponding p-value for a certain choice of mass and number of simulation steps, over a range of step sizes. The χ^2 values are obtained using 20 equally spaced bins over each dimension.

illustrate how combining MC^3 with advanced local samplers can mitigate individual weaknesses of a global Metropolis-Hasting sampler and a local sampler.

4.2. Multi-Channel Markov Chain Monte Carlo with Hamiltonian Local Update

Having introduced all basic sampling methods, the advantage of using an informed sampling method (HMC) as local update in MC^3 (HMC-MC^3) can now be analyzed. Varying the β-parameter of MC^3 is equivalent to changing the dominance of the local update relative to the global update, with $\beta = 1$ corresponding to only using the global update. Varying the parameter β close to 1 can therefore be interpreted as analyzing the performance improvements on a global update by introducing the additional local update. Since the local update alone (in the case of HMC) was already seen to be sub-optimal for distributions with multiple distinct peaks, the mixing can also be seen as improving on the local update which dominates for β close to 0. In summary the sample quality must be analyzed for $\beta \in [0, 1]$, where values close to the extremes

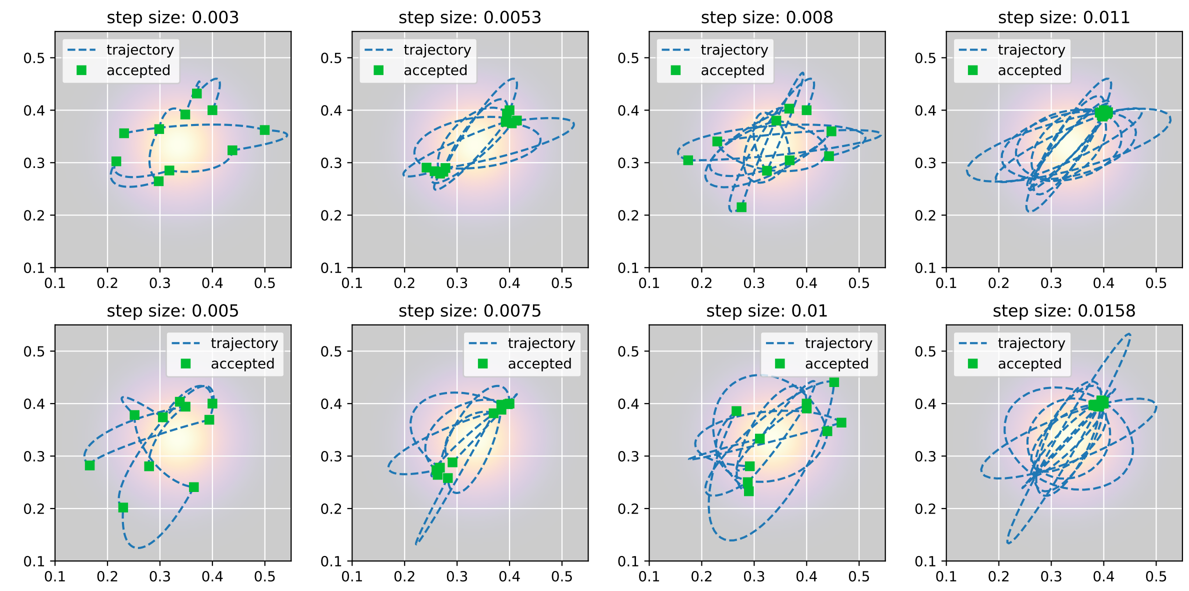

Figure 4.2.: Trajectories of an HMC update with a Gaussian target distribution ($\mu = (1/3, 1/3)^\intercal$ and $\sigma^2 = 0.1^2/2$), for masses of 1 (top row) and 2 (bottom row), and 40 simulation steps.

correspond to only using the local or global update, respectively.

4.2.1. Comparing Performance Improvements on the Camel Distribution

Following the previous examples, fig. 4.4 shows the sampling results for a two-dimensional Camel target distribution (with default parameters, eq. (2.11)) over the range of possible values for β. All data points are obtained as the average over 20 runs of the methods.

Three different global importance sampling distributions p_{IS} for the MC^3 are considered. The first distribution is a *perfect mapping*, being exactly the target distribution. This corresponds to a multi-channel configuration using as channels two Gaussian distributions at the centers with equal weights. The second multi-channel configuration for the *imperfect mapping* uses as channels the two Gaussian distributions with parameters from the importance sampling example in eq. (2.14). The final distribution p_{imp} in this case is obtained by executing the adaptive multi-channel Monte Carlo algorithm from section 2.4 for 1000 integration steps. This importance distribution most resembles a realistic sampling problem using e.g. the Sarge [16] algorithm as phase space mapping, with limited information about the exact peak locations and shapes. In addition p_{uniform}, a uniform distribution over the whole volume is used, which loosely corresponds to a RAMBO [17] phase space mapping and no information about the target distribution (see section 5.1).

As expected, the global Metropolis-Hasting sampler using the perfect mapping, showing as a straight line as it has no dependence on β, performs best and serves as reference for the optimal performance (the sample points are proposed directly according to the target distribution). The

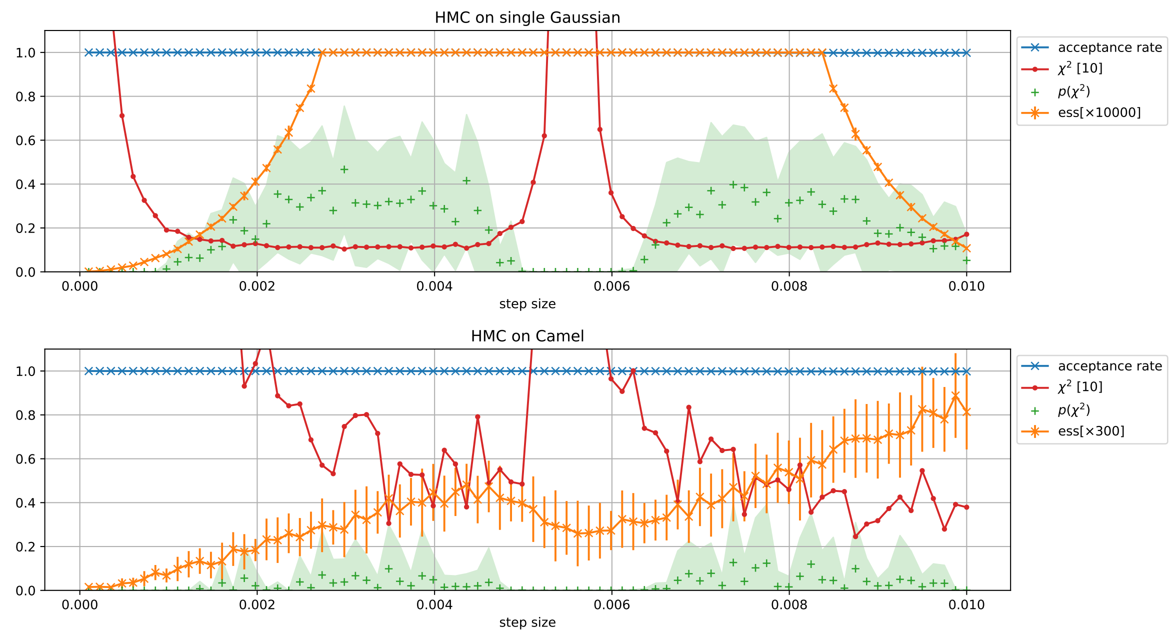

Figure 4.3.: ESS, χ^2 value, corresponding p-value, and acceptance rate measuring the sample quality of HMC on a Gaussian (top) and a Camel (bottom) distribution (both two-dimensional) as target, for a mass 1 and 40 simulation steps, over a range of step sizes. The Gaussian distribution has parameters $\mu = (1/3, 1/3)^\intercal$ and $\sigma^2 = 0.1^2/2$, for the Camel distribution the default values from eq. (2.11) are used. Each data point is the mean over 20 runs of the HMC sampler, the variance is shown for the ESS and the χ^2 p-value.

global Metropolis-Hasting sampler for the imperfect mapping (again showing as straight line) performs significantly worse but notably maintains a high ESS. This result corresponds with the currently established method to generate phase space events, and must be improved upon.

The basic MC^3 method with a local Gaussian distribution and an optimal variance of eq. 3.14 corresponds to the method introduced in [1]. The consistently decreasing χ^2 value relative to β implies the local update is (for this example) worse than using the an update with the global imperfect mapping on its own, and using the basic MC^3 algorithm leads to no improvement (for more and narrower peaks in the target distribution this will likely differ).

The two HMC-MC^3 configurations with step sizes 0.003 and 0.008 (chosen according to the analysis in sections 4.1.2 and 4.1.3) seem to perform equally, which indicates the likelihood of HMC crossing the gap between the peaks in the Camel distribution becomes irrelevant when using MC^3. For small values of β this method corresponds to just using HMC. By increasing β, both the χ^2 value and the ESS are improved. While there seems to be an optimal value for β regarding the sample's conformity with the target distribution (minimal χ^2), the ESS rises monotonically (the importance sampling update produces uncorrelated points). The minimum in the χ^2 plot can be understood as a compromise between using the HMC update, better suited to sample a Gaussian distribution, and the global importance sampling update, worse at sampling the Gaussian peaks but able to "jump" between them. In practice, a balance between producing an optimal ESS and χ^2 value, informed by the actual time efficiency of either and the desired sample quality must be found. For a realistic (instead of the proof of concept Python implementation done and used here) the ratio of the ESS and the computation time may be used as a measure.

A similar behavior for the ESS and χ^2 is visible for the HMC-MC^3 method using a uniform importance sampling distribution. However, the optimal χ^2 value as well as the corresponding p-value are considerably worse then the result obtained using $p_{\text{IS}} = p_{\text{imp}}$. The ESS increases for larger values of β but reaches a significantly smaller maximum, which is the value that would be obtained using only the global Metropolis-Hasting sampler with uniform proposal. The worse performance directly corresponds with less information being available about the target distribution.

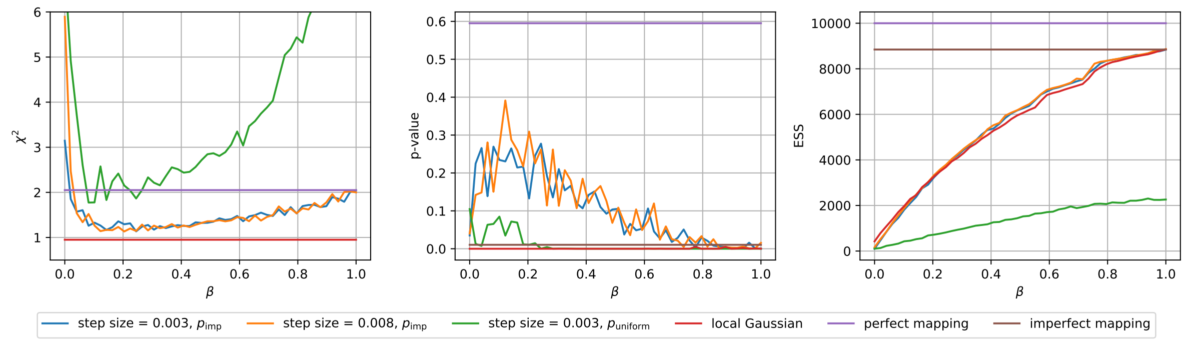

Figure 4.4.: Sampling results of a two-dimensional Camel distribution using the MC^3 algorithm in various configurations. The plots show the χ^2 (left), p-value (middle) and ESS (right) over $\beta \in [0, 1]$ for various configurations of MC^3, as well as global Metropolis-Hasting updates with perfect (target distribution) and imperfect (Camel distribution according to eq. (2.14)) proposals. The first two MC^3 configurations use the imperfect mapping as importance distribution p_{IS} and HMC as local update with step sizes 0.003 and 0.008. The third configuration uses HMC with step size 0.003 and a uniform distribution as p_{IS}. Finally, the basic MC^3 configuration using a local Gaussian proposal with optimal variance (eq. (3.14)) is displayed. The χ^2 values are obtained using 20 equally spaced bins over each dimension.

4.3. Extreme Learning Surrogate for the Potential Gradient

The HMC method in section 4.1 relies on the potential gradient of the target distribution. Neither the potential $V(x) = \log f(x)$ for a target distribution f, nor its gradient $\nabla V(x)$ are analytically known for applications in high energy particle physics. The function f is expensive to evaluate which makes a surrogate for the potential gradient desirable. The approach chosen here is to computationally *learn* the potential

$$V(x) = -\log f(x) \approx z(x), \tag{4.6}$$

in a way that a numerical expression for the gradient of z is immediately known and

$$\nabla V(x) \approx \nabla z(x). \tag{4.7}$$

The *extreme learning machine* (ELM) proposed in [18] can be used to find the optimal weights in a single-hidden layer feedforward neural network. The variant considered here is for one-dimensional output, and based on radial basis functions (specifically Gaussian) such that the final output function for an input value x^j (generally denoting a point in K-dimensional space) is

$$z(x^j) = \sum_{i=1}^{Z} v_i a_i(x^j; c^i, w^i) = \sum_{i=1}^{Z} v_i a_i \left(-\frac{\left\| x^j - c^i \right\|}{2(w^i)^2} \right). \tag{4.8}$$

The number of nodes is Z and each node i corresponds to a radial basis function a_i with a bias c^i and a width w_i. Each node may generally have a different output function a_i, but for the implementation and application here, all are set to be Gaussian: $a_i(y) = a(y) = \exp y$. Then, the gradient of the surrogate z is immediately given by

$$\frac{\partial z(x)}{\partial x_k} = \sum_{i}^{Z} v_i \left(-\frac{x_k - c^i_k}{(w^i)^2} \right) a(x^j; c^i, w^i). \tag{4.9}$$

The parameters w^i and c^i are chosen randomly in the initialization of the optimization process (the ranges and distributions for the parameters must be chosen for the specific input space).

The output matrix H for a set of input values x^j is defined as

$$H_{ij} = a(x^j; c^i, w^i), \tag{4.10}$$

such that the output using a certain weight vector v (which is to be optimized) is

$$z^j = \sum_{i}^{Z} H_{ij} v_i, \; z = Hv. \tag{4.11}$$

The training problem is defined by a set of data x^j with known output $T = (f(x^1), \ldots, f(x^t))$ such that $Hv = T$. If the number of training data t is smaller or equal the number of nodes Z, the problem can be exactly solved for almost all parameter values [18]. Generally it will be $Z < t$ and H is not invertible. The optimal output weight vector is chosen using the pseudo inverse $H^\intercal$ (Moore-Penrose generalized inverse), which minimizes the root-mean-square differences between the learned outputs Hv and the training set T.

Fig. 4.5 gives a qualitative comparison of the optimization results for an ELM using 500 Gaussian nodes as described above, approximating the potential for a Camel distribution (eq. (2.10), the exact gradient is given in eq. (2.12)). The root-mean-square differences to the surrogate (using 100 000 uniformly spaced test points) for the potential is 0.0509, and for the gradient 82.7. While this confirms the overall functionality of the method, fig. 4.6 shows the root-mean-square differences (RMS) of the surrogate to the actual Camel distribution (computed using other points than the training data) over different number of training points. A convergent behavior is visible, the RMS decreases as expected with more training points used until leveling off, which corresponds to the best possible configuration (given the network of basis functions) being reached. The RMS is notably higher for the gradient, which is to be expected as the ELM optimizes the network to match the potential.

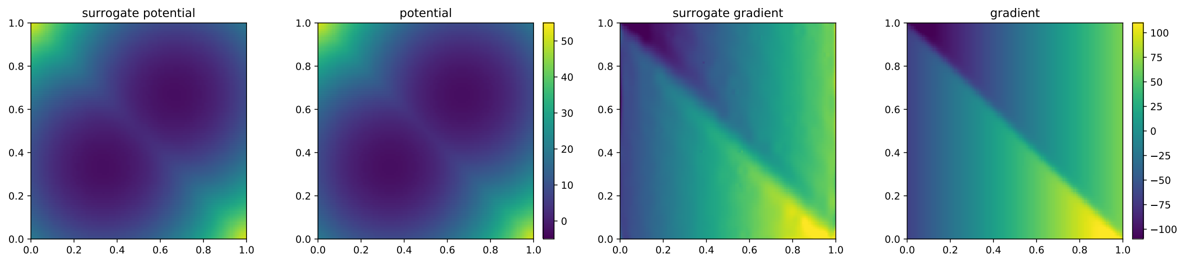

Figure 4.5.: Qualitative comparison of the potential and potential gradient of the Camel distribution (eq. (2.10), (2.11)) with the ELM surrogates. The ELM uses 100 or 500 Gaussian nodes with widths in $[0.001, 0.5]$ and centers in $[0, 1]^2$, and was trained with 100 000 points randomly chosen within the volume.

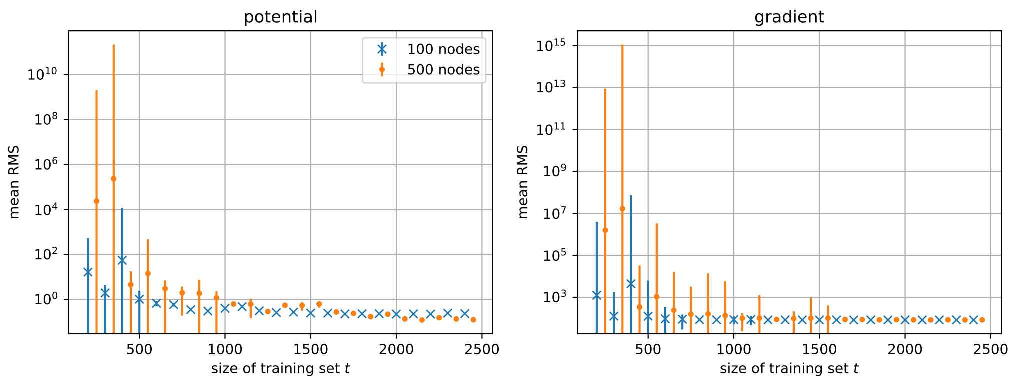

Figure 4.6.: Average root-mean-square of differences to the exact function (using 100 000 uniformly spaced points as test) for 20 training runs (and the correspondingly obtained variance) using the ELM to obtain the potential to the Camel distribution (eq. (2.10), (2.11)).

The reason for learning the potential and implicitly obtaining the gradient is that training data for the potential can easily be obtained. The method of an ELM can be applied within the MC^3 framework without any additionally needed function evaluations, if the function values used in the integration phase can be reused. The computation cost of the ELM is dominated by obtaining the pseudo inverse of the output matrix.

Especially when using the ELM in combination with HMC, distributions with significant areas of zero probability may lead to problems as there are training points in those regions (the logarithm goes to infinity for values going to zero). If the distribution continuously approaches zero, the potential would be learned to go to infinity and the resulting HMC dynamics would usually be contained in the sensible areas. If the distribution, however (e.g. due to border conditions) instantaneously drops to zero, the lack of data means the ELM surrogate may assume any behavior which may make HMC inefficient (see section 5.2). It is important to note that a bad surrogate only leads to bad sampling performance but does not impact the correct limit distribution of HMC. The issue of borders, as mentioned in section 4.1.1 can be addressed by using e.g. Spherical HMC [9].

4.3.1. No-U-Turn Sampler

The main issue of HMC-MC^3 is the large number of parameters that must be tuned with respect to the given target distribution. The No-U-Turn sampler introduced in [6] is an adaptive variant of the HMC algorithm which tunes the step width using a dual averaging scheme [15] and chooses an optimal number of steps for each proposal. While a detailed description of the sampler is beyond the scope of this introduction, the basic concept may be outlined.

The advantage of HMC methods is that far-away states are proposed with close to perfect acceptance rate, reducing the random-walk behaviour and lag-autocorrelation compared to a basic local Metropolis-Hasting algorithm. Based on this, the No-U-Turn sampler chooses the step width for each proposal such that the proposed state is as far as possible from the initial state. This is done by simulating the trajectory (in both directions to maintain reversibility and detailed balance) up to a point where the next step would have a backward component in the direction between initial and that point.

The basic algorithm is, however, problematic if applied to the Camel distribution considered in the previous sections. The basic structure of either peak is Gaussian, for which the Hamiltonian trajectories are ellipses. Based on the outline of the algorithm, the number of steps would always be selected such that the trajectories stop at a half turn (or an integer multiple plus a half turn) of the ellipsis. This trajectory, corresponding to the second plot in fig. 4.2, leads to arbitrarily slow convergence, as was seen section 4.1.2. Running an implementation of the No-U-Turn sampler[1] confirms this behavior. Even if this kind of “bad” convergence could be mitigated, running the No-U-Turn sampler on the Camel distribution, used here as an artificial toy problem, would not lead to further insights as the previous analysis corresponds to a manual optimization. While the No-U-Turn sampler leads to no further insight for the Camel distribution, the sampling result for the physical example considered in section 5.2 serves as indication that it can generally be used to eliminate the need for manual tuning.

[1]implemented by Timo Janßen

5. Sampling Physical Distributions over Phase Space

In the previous sections, Monte Carlo sampling and integration methods were discussed generally and analyzed using an artificial probability distribution. The developed methods, specifically HMC and MC^3, will now be applied to a real physical process.

For n final state particle, the phase space is naively $4n$-dimensional (one four-momentum for each particle). Based on this alone, the sampling method would have to use $\mathbb{R}^n$ as sample space. However, due to the on-shell condition for the final state particles and four-momentum conservation, the sample space can be greatly confined ($f = 0$ for areas where these constraints are not satisfied) and the dimension can be reduced to $3n - 4$. For the Monte Carlo methods it is vital to use the mapping with reduced number of dimensions since trying to sample from a manifold within a higher dimensional hyper-cube would fail (without building the constraint of the manifold into the sampling method, the likelihood of randomly choosing a point with non-zero probability is zero). Note that the manifold is associated with a δ-distribution such that the integration over $\mathbb{R}^n$-phase space is non-zero.

To obtain the differential cross section f the Python interface of the Sherpa [10] framework is used. The configuration files used in the following are available together with the rest of the Python source code used here[1].

5.1. RAMBO Phase Space Mapping

The RAMBO-algorithm introduced in [17] maps values from the hyper-cube $[0, 1]^{4n}$ to physical four-momenta with a fixed center of mass energy [3]. The approach is to first consider the phase space of n massless four-momenta, not constrained by momentum conservation and then Lorentz boost and re-scale them into physical four-momenta. The algorithm for the center of mass energy P, which will not be further derived here, consists of two steps as described in [3] and included here for completeness:

1. n massless four-momenta q_i^μ are generated independently with isotropic angular distribu-

[1]`https://github.com/mathisgerdes/hep-monte-carlo/releases/tag/v1.1`

tion the energy component q_i^0 according to the density $p(q_i^0) = e^{-q_i^0}\,\mathrm{d}q_i^0$. For $u_i \in [0,1]^4$:

$$\begin{aligned} c_i &= 2u_i^1 - 1, & \phi_i &= 2\pi u_i^2, & q_i^0 &= -\log\left(u_i^3 u_i^4\right) \\ q_i^x &= q_i^0\sqrt{1-c_i^2}\cos\phi_i, & q_i^y &= q_i^0\sqrt{1-c_i^2}\sin\phi_i, & q_i^z &= q_i^0 c_i. \end{aligned} \tag{5.1}$$

2. The q_i^μ are Lorentz and scaling transformed:

$$p_i^0 = x\left(\gamma q_i^0 + \vec{b}\cdot\vec{q}_i\right), \qquad \vec{p}_i = x\left(\vec{q}_i + \vec{b}q_i^0 + a\left(\vec{b}\cdot\vec{q}_i\right)\vec{b}\right), \tag{5.2}$$

using the variables

$$Q^\mu = \Sigma_{i=1}^n q_i^\mu, \qquad M = \sqrt{Q^2}, \qquad \vec{b} = -\frac{1}{M}\vec{Q}, \tag{5.3}$$

$$\gamma = \frac{Q^0}{M} = \sqrt{1+\vec{b}^2}, \qquad a = \frac{1}{1+\gamma}, \qquad x = \frac{\sqrt{P^2}}{M}. \tag{5.4}$$

Points generated this way have an associate weight (corresponding to a uniform probability density for the four-momenta) of

$$w_0 = (2\pi)^{4-3n}\left(\frac{\pi}{2}\right)^{n-1}\frac{(P^2)^{n-2}}{\Gamma(n)\Gamma(n-1)}. \tag{5.5}$$

5.2. Electron-Positron Annihilation to Quark-Antiquark Pair

Fig. 5.1 shows the differential cross section of $e^+e^- \to q\bar{q}$ obtained via Sherpa and using RAMBO on diet as mapping (such that it can be displayed in two dimensions). The center of mass energy used is $P = 100\,\mathrm{GeV}$. For the integration and sampling, the distribution/integrand f will refer to the differential cross section composed with the RAMBO on diet mapping so that $f : [0,1]^2 \to \mathbb{R}_{\geq 0}$. The probability distribution only varies in one dimension, which indicates the probability is only dependent on one out of the two degrees of freedom. This corresponds to the process being independent of the azimuthal angle ϕ of the spherical coordinate angles ϕ and θ describing a $2 \to 2$ scattering.

In the first step, the differential cross section is integrated using importance sampling and RAMBO on diet as mapping (including the corresponding weight as probability density). For 100 000 integration steps, the result for the total cross section is

$$206.6 \pm 0.3\,\mathrm{pb}, \tag{5.6}$$

which is consistent with the result obtained by Sherpa (using Rambo).

Since the distribution is relatively flat, a simple Metropolis sampler with the RAMBO on diet mapping as proposal can be used to generate events. An exemplary run of 10 000 sample points gives an acceptance rate of 78 % and

$$\chi^2 = 1.56, \qquad p(\chi^2) = 0.00015, \qquad ESS = 5\,595. \tag{5.7}$$

To obtain a reasonable χ^2 value, the binning was done over the $[0,1]^2$ hyper-cube, using the same mapping as for the sampling. In the following sample analyses, the binning is fixed to 20 bins in each dimension. The variance and mean of the distribution, needed for the ESS, was obtained from a similar Metropolis sample of 100 000 events.

In order to test HMC methods on this distribution, first an expression for the potential gradient must be obtained. This is done using the ELM, for which a total RMS error for the potential reaches 0.00038 using 50 nodes and a training set of size 1000. A trajectory generated by an HMC run using this potential gradient is visible in fig. 5.1. It is clearly visible that, since the HMC algorithm does not have the border constraint of the unit hyper-cube built-in, and the potential may assume any behavior in regions it is not trained on, the performance of a pure HMC sampler will suffer from simulations moving out of the $[0,1]^2$ area. An exemplary run using the same parameters as for the shown trajectory (mass 1, step size of 0.01 and 40 simulation steps) gives an acceptance rate of 36 % and

$$\chi^2 = 5.44, \qquad p(\chi^2) = 1 \times 10^{-61}, \qquad ESS = 923. \tag{5.8}$$

This is significantly worse than the uninformed, global Metropolis sampler because the HMC dynamics frequently moves out of the unit hyper-cube, whereas the previously used Metropolis sampler only proposes states within the hyper-cube.

In order to effectively use HMC for distributions that instantaneously drop to zero due to border constraints, Spherical HMC can be used [9]. Instead of simulating the Hamiltonian dynamics directly on the unit hyper-cube, the space is mapped onto a higher dimensional sphere. Using this method[2] with the same parameters as before, an acceptance rate of 99.97 % is reached with

$$\chi^2 = 2.77, \qquad p(\chi^2) = 5 \times 10^{-20}, \qquad ESS = 10\,000, \tag{5.9}$$

which is a significant improvement over the basic HMC method. The method was not optimized for the parameters of the step size and step width and better performance (especially regarding faster converges) may be achievable. When using the spherical mapping, it was observed that for large step sizes points close to the edges were unlikely to be sampled. This may be due to a relative stretching between the sample space (the hyper-cube) and the sphere, which is especially important for the distribution used here with significant areas of high probability close to the edges, or an unexpected behavior of the resulting Hamiltonian dynamics.

A proof of concept run of a No-U-Turn sampler including the Spherical HMC mapping[2] leads

[2]implemented by Timo Janßen

to an acceptance rate of 92% and

$$\chi^2 = 4.67, \qquad p(\chi^2) = 4 \times 10^{-48}, \qquad ESS = 10\,000. \tag{5.10}$$

While this indicates the implementation is functional, the performance seems worse than for the basic Spherical HMC. A qualitative inspection of a scatter plot shows notably few sample points close to the edges. This leads to believe the step size may be optimized to a value where the spherical mapping impairs the dynamics as was suspected earlier. Further analysis of the algorithm as well as the sampling results is needed to verify this hypothesis.

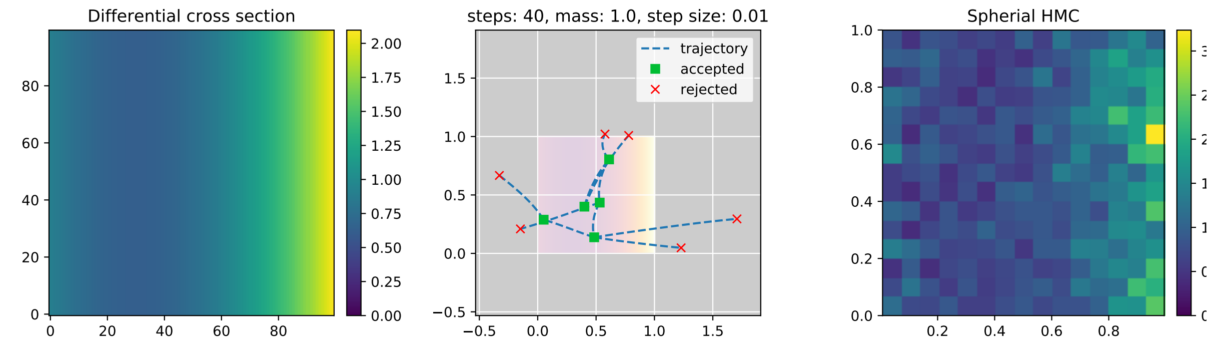

Figure 5.1.: Normalized differential cross section (right) for the process $e^+e^- \to q\bar{q}$, obtained via Sherpa and displayed here using the RAMBO on diet mapping. The HMC generated trajectories (middle) are obtained for a step size of 0.01, a mass of 1 and 40 simulation steps; the required potential gradient was obtained using the ELM with Gaussian radial basis functions. The right plot shows the sample distribution obtained using Spherical HMC and 10 000 sample points.

6. Summary of Results

The analysis of HMC-MC^3, in comparison to the basic MC^3 method and a global Metropolis Hasting sampler, shows a potentially significant increase in sampling efficiency for sufficiently complex distributions. The performance improvement depends, however, on the tuning of at least the β-parameter and relies on a numerical approximation of the potential gradient. Using an ELM surrogate for the potential gradient proves functional for the analyzed distributions.

The ESS and the χ^2 value are generally optimized for different values of β and a compromise must be made, based on the distribution specific quality of the global and local updates. For a "production" implementation, the efficiency of the HMC-MC^3 configuration can be measured using the ratio of the ESS and the needed computational time (including the optimization phase for the surrogate). This makes the method comparable to basic Metropolis-Hasting updates, taking into account additional costs due to HMC simulations. An analytical derivation of the computational cost (using the number of function evaluations) cannot be sensibly made, since the surrogate can be learned without additional function evaluations and, using the surrogate, the simulation of the Hamiltonian dynamics also does not need to evaluate the target distribution. This makes the HMC method, in terms of function evaluations, equally expensive as a simple Metropolis-Hasting update. Besides the efficiency of the chosen configuration (given by the effective sample size produced per computation time), the χ^2 value (and corresponding p-value) must be taken into account to confirm the sampling method converges to the target distribution reasonably well, within the desired number of sample points.

If the target distribution is relatively flat, as in section 5.2, the HMC dynamics close to the edges of the sample space becomes relevant and must be carefully handled. Spherical HMC, only shortly mentioned here, seems a promising resolution of this issue, but must be further analyzed.

6.1. Outlook

One of the main drawbacks of the HMC-MC^3 combination presented here is the high number of parameters which must be chosen and tuned for individual distributions. While the β-parameter must be chosen based on the mapping quality, the step size and number of steps used in the HMC algorithm can be adaptively tuned using more advanced variants such as the No-U-Turn sampler. While the implementation (not personally implemented) shortly tested in section 5.2 seems functional, further analysis is required to confirm an optimal configuration is reached and explore possible drawbacks in performance. Specifically, the handling of border constraints must be studied and the combination of the Spherical HMC mapping with the No-U-Turn sampler must be validated.

RAMBO, used as physical phase-space mapping here, is fundamentally a single-channel method and would, for differential cross sections with significant peaks, lead to slow sample convergence (for all sampling methods and the integration phase). An alternative method is the Sarge algorithm, introduced in [16].

While the ELM surrogate for the potential gradient proved sufficient here, it must, especially regarding the computational cost, be further analyzed for more realistic problems. The main bottleneck in obtaining the surrogate is the computation of the pseudo inverse of the output matrix. This may be improved upon by using iterative optimization algorithms, which can also be used to improve the weight vector using individual points (for example during an initial burn-in phase of the MC^3 sampler).

A. Choice and Estimator for the Bin Width

The computation of the χ^2 value relies on a binning of the sample space. Theoretically, the binning can be arbitrarily chosen. The requirement of a minimal number of sample points restrains how small the bin width can reasonably be. Furthermore, choosing the bin size too large would miss features of the sample distribution.

In all practical applications here, the binning is chosen to be uniform in each dimension. While there exist multiple estimators for a good bin width in one dimension, such as the Freedman Diaconis Estimator [19]

$$h = 2\, IQR\, N^{-1/3}, \tag{A.1}$$

where IQR is the interquartile range and N the number of sample points, estimators for more than one dimensions are less common. Based on the proportionality to IQR, which increases the bin width relative to the scale of the distribution, and the scaling $N^{-1/(2+K)}$ in the dimensionality K [20], the estimator used here is

$$h = 2\, IQR\, N^{-1/(2+K)}. \tag{A.2}$$

For applications comparing the efficiency of multiple methods or various configuration, the binning is fixed to a number based on this estimator. If no choice of binning is mentioned, the result of this estimator is used for the bin width.

Bibliography

[1] Kevin Kröninger, Steffen Schumann, and Benjamin Willenberg. "(MC)**3 – a Multi-Channel Markov Chain Monte Carlo algorithm for phase-space sampling". In: *Comput. Phys. Commun.* 186 (2015), pp. 1–10. DOI: `10.1016/j.cpc.2014.08.024`. arXiv: `1404.4328 [hep-ph]`.

[2] Andy Buckley et al. "General-purpose event generators for LHC physics". In: *Phys. Rept.* 504 (2011), pp. 145–233. DOI: `10.1016/j.physrep.2011.03.005`. arXiv: `1101.2599 [hep-ph]`.

[3] Stefan Weinzierl. "Introduction to Monte Carlo methods". In: *ArXiv High Energy Physics - Phenomenology e-prints* (2000). arXiv: `hep-ph/0006269 [hep-ph]`.

[4] Steve Brooks et al. *Handbook of Markov Chain Monte Carlo.* Chapman and Hall/CRC Handbooks of Modern Statistical Methods. Chapman and Hall/CRC, 2011.

[5] R. M. Neal. "MCMC using Hamiltonian dynamics". In: *ArXiv e-prints* (June 2012). arXiv: `1206.1901 [stat.CO]`.

[6] M. D. Hoffman and A. Gelman. "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo". In: *ArXiv e-prints* (Nov. 2011). arXiv: `1111.4246 [stat.CO]`.

[7] H. Strathmann et al. "Gradient-free Hamiltonian Monte Carlo with Efficient Kernel Exponential Families". In: *ArXiv e-prints* (June 2015). arXiv: `1506.02564 [stat.ML]`.

[8] S. Duane et al. "Hybrid Monte Carlo". In: *Phys. Lett.* B195 (1987), pp. 216–222. DOI: `10.1016/0370-2693(87)91197-X`.

[9] S. Lan, B. Zhou, and B. Shahbaba. "Spherical Hamiltonian Monte Carlo for Constrained Target Distributions". In: *ArXiv e-prints* (Sept. 2013). arXiv: `1309.4289 [stat.CO]`.

[10] T. Gleisberg et al. "Event generation with SHERPA 1.1". In: *JHEP* 02 (2009), p. 007. DOI: `10.1088/1126-6708/2009/02/007`. arXiv: `0811.4622 [hep-ph]`.

[11] G. Peter Lepage. "A New Algorithm for Adaptive Multidimensional Integration". In: *J. Comput. Phys.* 27 (1978), p. 192. DOI: `10.1016/0021-9991(78)90004-9`.

[12] Ronald Kleiss and Roberto Pittau. "Weight optimization in multichannel Monte Carlo". In: *Comput. Phys. Commun.* 83 (1994), pp. 141–146. DOI: `10.1016/0010-4655(94)90043-4`. arXiv: `hep-ph/9405257 [hep-ph]`.

[13] William G. Cochran. "The χ^2 Test of Goodness of Fit". In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 315–345. ISSN: 00034851.

[14] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method”. In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. ISSN: 00034851. URL: `http://www.jstor.org/stable/2236626`.

[15] Yurii Nesterov. “Primal-dual subgradient methods for convex problems”. In: *Mathematical Programming* 120.1 (Aug. 2009), pp. 221–259. ISSN: 1436-4646. DOI: `10.1007/s10107-007-0149-x`.

[16] Petros D. Draggiotis, Andre van Hameren, and Ronald Kleiss. “SARGE: An Algorithm for generating QCD antennas”. In: *Phys. Lett.* B483 (2000), pp. 124–130. DOI: `10.1016/S0370-2693(00)00532-3`. arXiv: `hep-ph/0004047 [hep-ph]`.

[17] R Kleiss, W. J. Stirling, and S. D. Ellis. “A new Monte Carlo treatment of multiparticle phase space at high energies”. In: *Computer Physics Communications* 40.2-3 (June 1986), pp. 359–373. ISSN: 00104655. DOI: `10.1016/0010-4655(86)90119-0`.

[18] Guang Bin Huang, Qin Yu Zhu, and Chee Kheong Siew. “Extreme learning machine: Theory and applications”. In: *Neurocomputing* 70.1-3 (2006), pp. 489–501. ISSN: 09252312. DOI: `10.1016/j.neucom.2005.12.126`. arXiv: `1311.4555`.

[19] David Freedman and Persi Diaconis. “On the histogram as a density estimator: L2 theory”. In: *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 57.4 (Dec. 1981), pp. 453–476. ISSN: 1432-2064. DOI: `10.1007/BF01025868`.

[20] David W. Scott. *Multivariate Density Estimation.* Wiley Series in Probability and Statistics. Hoboken, NJ: John Wiley & Sons, Inc, Mar. 2015. ISBN: 9781118575574. DOI: `10.1002/9781118575574`.