

# Computational Statistics

## Project Report

20th March 2020

### 1 Introduction

The Markov Chain Monte Carlo (MCMC) sampling tries to investigate properties of distributions by drawing random samples. This is a sequential process, where the drawing of a random sample at any given time depends only on its immediate predecessor. In this project a variant of the MCMC algorithm from the paper [Lau and Krumscheid, 2019] is reimplemented and an attempt is made to reproduce the results of the experiments. The paper under consideration builds on the ideas of [Metropolis et al., 1953, Schmeiser, 1992, Liu et al., 2000] and [Yang et al., 2019]. The source code under MIT license for this project can be found in the GitHub repository

`github.com/rinkwitz/Adaptive_Plateau_MCMC`.

Since the simulation of Markov chains is very time-consuming, you can download already simulated MCMC runs under the link

`drive.google.com/file/d/10zUR6-axu2M8hrUERXLregBqmXdpoS-Y/view?usp=sharing`.

### 2 Adaptive Component-wise Multiple-Try Metropolis Algorithm

The core algorithm of the paper [Lau and Krumscheid, 2019] consists of a MCMC algorithm which fulfills three properties. First, the algorithm suggests several suggestions

from different plateau distributions during sampling. This happens independently for all components of a sample. Lastly, plateau distributions adapt their shape depending on the frequency of the accepted sample proposals.

## 2.1 Plateau Proposal Distributions

The MCMC algorithm in [Lau and Krumscheid, 2019] uses *non-overlapping plateau proposal distributions* for sampling. The underlying probability density function  $f$  is a combination of the density of a uniform distribution with exponential decaying tails. Here the density  $f$  is constant around a mean value  $\mu$  in the closed interval  $[\mu - \delta, \mu + \delta]$  with  $\delta > 0$ . Outside this interval the distribution follows an exponential decay with its tail width determined by  $\sigma_i > 0$ , depending on whether you are on the left or right side of the interval  $[\mu - \delta, \mu + \delta]$ . If you define an unnormalized density function

$$\tilde{f}(y; \mu, \delta, \sigma_1, \sigma_2) = \begin{cases} \exp\left(-\frac{1}{2\sigma_1^2}[y - (\mu - \delta)]^2\right) & , y < \mu - \delta \\ 1 & , \mu - \delta \leq y \leq \mu + \delta \\ \exp\left(-\frac{1}{2\sigma_2^2}[y - (\mu + \delta)]^2\right) & , y > \mu + \delta \end{cases}$$

and then calculate the following integral

$$\begin{aligned} C(\delta, \sigma_1, \sigma_2) &= \int_{-\infty}^{\infty} \tilde{f}(y; \mu, \delta, \sigma_1, \sigma_2) dy \\ &= \int_{-\infty}^{\mu - \delta} \exp\left(-\frac{1}{2\sigma_1^2}[y - (\mu - \delta)]^2\right) dy + \int_{\mu - \delta}^{\mu + \delta} 1 dy + \int_{\mu + \delta}^{\infty} \exp\left(-\frac{1}{2\sigma_2^2}[y - (\mu + \delta)]^2\right) dy \\ &= \frac{\sqrt{2\pi\sigma_1^2}}{2} + 2\delta + \frac{\sqrt{2\pi\sigma_2^2}}{2} \end{aligned}$$

as the sums of 2 half gaussian integrals and one integral over a constant function, then the normalized probability density function is  $f(y; \mu, \delta, \sigma_1, \sigma_2) = C(\delta, \sigma_1, \sigma_2)^{-1} \tilde{f}(y; \mu, \delta, \sigma_1, \sigma_2)$  [Lau and Krumscheid, 2019]. Using  $f$  you can define the plateau probability density distributions  $T_{j,k}$ ,  $j \in \{1, \dots, M\}$  for the trial proposals of the  $k$ -th component as

$$T_{j,k}(x, y) = \begin{cases} f(y; x, \delta_1, \sigma, \sigma) & , j = 1 \\ \frac{1}{2}f(y; x - (2j - 3)\delta - \delta_1, \delta, \sigma, \sigma) + \frac{1}{2}f(y; x + (2j - 3)\delta + \delta_1, \delta, \sigma, \sigma) & , j = 2, \dots, M - 1 \\ \frac{1}{2}f(y; x - (2M - 3)\delta - \delta_1, \delta, \sigma_0, \sigma) + \frac{1}{2}f(y; x + (2M - 3)\delta + \delta_1, \delta, \sigma, \sigma_1) & , j = M \end{cases}$$

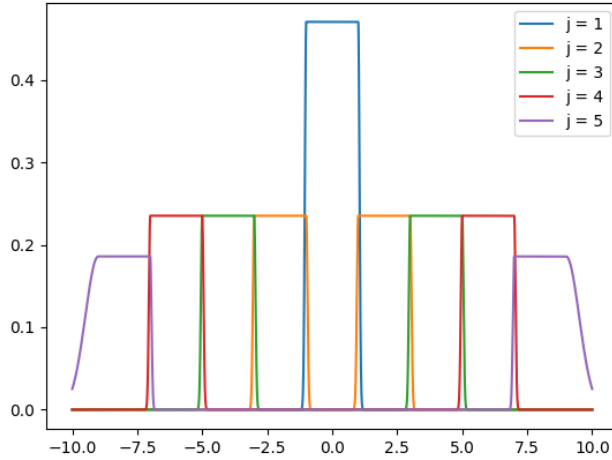


Figure 1: Trial proposal probability density distributions for  $M = 5$

with values  $\delta_1, \delta, \sigma, \sigma_0, \sigma_1 > 0$ . In figure 1 you can see the trial proposal probability density distributions for the parameters  $M = 5, \delta_1 = \delta = 1, \sigma = 0.05, \sigma_0 = \sigma_1 = 0.5$ . One can see that the distributions overlap only at their exponential decaying tails. The outer tails decrease with the larger  $\sigma_0, \sigma_1$  values shallower than the remaining tails. In the paper [Lau and Krumscheid, 2019], the authors consistently use the values  $\delta = \delta_1 = 2, \sigma = 0.05, \sigma_0 = \sigma_1 = 3$ .

## 2.2 Component-wise Multiple-Try Metropolis

The component-wise multiple-try Metropolis algorithm [Lau and Krumscheid, 2019, Algorithm 1], which forms the basis of the sampling procedure, starts from a starting position  $x_0 \in \mathbb{R}^d$ . For each MCMC realization  $x_n$  with  $n \in \{1, \dots, N\}$  the following procedure is used for each one of the  $d$  components. If  $x = (x_1, \dots, x_d)$  is the last sampled candidate of the MCMC algorithm, then the algorithm proposes trials  $z_j$  for  $i = 1, \dots, M$  by sampling it from the distributions  $T_{j,k}(x_k, \cdot)$ . In my reimplementation I use a rejection sampling procedure [Peng, 2018]. For this purpose I use an uniform

distribution to generate the samples over the following intervals

$$I_j = \begin{cases} [x_k - \delta_1 - t_1, x_k + \delta_1 + t_1] \\ \text{for } j = 1 \text{ with } t_1 = \sqrt{-2\sigma^2 \log(0.0001C(\delta_1, \sigma, \sigma))}, \\ \\ [x_k - 2(j+1)\delta - \delta_1 - t_2, x_k - 2j\delta - \delta_1 + t_2] \cup [x_k + 2j\delta + \delta_1 - t_2, x_k + 2(j+1)\delta - \delta_1 + t_2] \\ \text{for } j = 2, \dots, M-1 \text{ with } t_2 = \sqrt{-2\sigma^2 \log(0.0002C(\delta, \sigma, \sigma))}, \text{ or} \\ \\ [x_k - 2(M+1)\delta - \delta_1 - t_{32}, x_k - 2M\delta - \delta_1 + t_{31}] \cup [x_k + 2M\delta + \delta_1 - t_{31}, x_k + 2(M+1)\delta + \delta_1 + t_{32}] \\ \text{for } j = M \text{ with } t_{31} = \sqrt{-2\sigma^2 \log(0.0002C(\delta, \sigma, \sigma_0))}, t_{32} = \sqrt{-2\sigma_0^2 \log(0.0002C(\delta, \sigma, \sigma_0))}. \end{cases}$$

These intervals enable effective sampling in the range of  $T_{j,k}(x_k, \cdot)$  where the probability density function is greater than 0.0001. To do this, one samples a  $u \sim U(0, 1)$  and a  $y \sim U(I_i)$  until

$$u < \frac{T_{j,k}(x_k, y)}{c|I_i|}$$

is fulfilled where  $|I_j|$  is the width of the used interval  $I_j$ ,  $g_j$  is the probability density function of the uniform distribution over  $I_j$  and  $c_j$  has the following values situation dependent

- $j = 1$  :  $c_1 = \sup_{y \in I_1} \frac{T_{1,k}(x_k, y)}{g_1(y)} = \frac{|I_1|}{C(\delta_1, \sigma, \sigma)}$ ,
- $j = 2$  :  $c_j = \sup_{y \in I_j} \frac{T_{j,k}(x_k, y)}{g_j(y)} = \frac{|I_j|}{2C(\delta, \sigma, \sigma)}$ , and
- $j = M$  :  $c_M = \sup_{y \in I_M} \frac{T_{M,k}(x_k, y)}{g_M(y)} = \frac{|I_M|}{2C(\delta, \sigma, \sigma_0)}$ .

Afterwards the weights associated with the trials are

$$w_{j,k} = \pi((z_j; x_{[-k]}))T_{j,k}(x_k, z_j)\lambda_{j,k}(x_k, z_j), \quad \text{for } j = 1, \dots, M$$

are calculated whereby  $(z; x_{[-i]}) \in \mathbb{R}^d$  denotes the vector, which is identical to  $x$  in all entries except the  $i$ -th one where it takes on the value  $z$ . In addition, the non-negative and symmetric function

$$\lambda_{j,k}(x, y) = |y - x|^{2.5}$$

is used. Here the authors of [Lau and Krumscheid, 2019] refer to the results of [Yang et al., 2019]. Thus, suggestions  $(z_j; x_{[-k]})$  have a high weight, which have a high probability with regard to the target distribution  $\pi$ , whose new proposal  $z_j$  for the  $k$ -th component regarding  $T_{j,k}(x_k, \cdot)$  is very likely and where the distance function  $\lambda_{j,k}$  is far enough away from  $x_k$ . Proportionally to the weights  $w_{1,k}, \dots, w_{M,k}$  a  $y \in \{z_1, \dots, z_M\}$  is then randomly drawn and for  $j = 1, \dots, M - 1$  the algorithm samples  $x_j^* \sim T_{j,k}(y, \cdot)$ . Finally one calculates

$$\alpha = \min \left\{ 1, \frac{w_{1,k}(z_1, x) + \dots + w_{M,k}(z_M, x)}{w_{1,k}(x_1^*, (y; x_{[-k]})) + \dots + w_{M-1,k}(x_{M-1}^*, (y; x_{[-k]})) + w_{M,k}(x_k, (y; x_{[-k]}))} \right\}$$

and with that probability  $\alpha$  the algorithm accepts the new proposal and sets  $x_n = (y; x_{[-k]})$ . Otherwise the algorithm keeps the old proposal and sets  $x_n = x$ .

## 2.3 Adaption of Proposal Distributions

In [Lau and Krumscheid, 2019] it is proposed to adapt the widths  $\delta$  and  $\delta_0$  of the plateau distributions. In my implementation every  $L$  iterations it measures how high the number of selected suggestions  $c_{j,k}$  from the plateau distributions  $T_{j,k}$  are in this interval. If the middle plateau distribution is called much more than the average, that is  $c_{j,k} > L\eta_1$  with  $\eta_1 \in (0, 1)$ , then the algorithm assumes that the plateaus are too wide and halves  $\delta$  and  $\delta_1$  accordingly for the following iterations. If, on the other hand, the outermost plateau distributions are selected much more than the average, in especially  $c_{M,k} > \eta_2 L$  with  $\eta_2 \in (0, 1)$ , then the algorithm assumes that the plateaus are too small and doubles  $\delta$  and  $\delta_1$ . The adaptations only take place with a constantly decreasing probability of  $\max(0.99^{n-1}, 1/\sqrt{n})$ . This implementation of the adaptation procedure is based on [Lau and Krumscheid, 2019, Algorithm 2].

# 3 Experiments and Results

## 3.1 Performance Measures

### 3.1.1 Integrated Autocorrelation Times

The authors in the paper [Lau and Krumscheid, 2019] use two performance measures to assess the effectiveness of the implemented algorithm. One of these is the integrated autocorrelation times (ACT), which is measured in  $R$  MCMC simulations with  $N$  steps for  $K$  components. Let in the following be  $X_t^{(r)} = (X_{t,1}^{(r)}, \dots, X_{t,K}^{(r)})$  the result of the  $r$ -th

MCMC simulation at step  $t$ , where  $r \in \{1, \dots, R\}$  and  $t \in \{1, \dots, N\}$ . First of all, one estimates the normalized autocorrelation  $\hat{p}_{\tau,k}^{(r)}$  by means of

$$\hat{c}_{\tau,k}^{(r)} = \frac{1}{N-\tau} \sum_{i=1}^{N-\tau} (X_{i,k}^{(r)} - \bar{X}_k^{(r)})(X_{i+\tau,k}^{(r)} - \bar{X}_k^{(r)}) \text{ and}$$

$$\hat{p}_{\tau,k}^{(r)} = \frac{\hat{c}_{\tau,k}^{(r)}}{\hat{c}_{0,k}^{(r)}}$$

where  $\bar{X}_k^{(r)} = 1/N \sum_{i=1}^N X_{i,k}^{(r)}$  is the arithmetic mean of the  $k$ -th component in the  $r$ -th simulation. Afterwards one approximates the integrated autocorrelation times as

$$\hat{\tau}_k^{(r)} = 1 + 2 \sum_{\tau=1}^M \hat{p}_{\tau,k}^{(r)}$$

[Foreman-Mackey, 2020] where  $M \ll N$ . To determine  $M$  the implementation uses a *initial monotone sequence estimator* as introduced by [?, geyer1992] where  $M$  is the largest natural number, so that  $\hat{p}_{\tau,k}^{(r)} > 0$  applies to all  $\tau \in \{1, \dots, 2M+1\}$  and the subsequence  $(\hat{p}_{2\tau,k}^{(r)} + \hat{p}_{2\tau+1,k}^{(r)})_{\tau=1,\dots,M}$  is monotone [Schmeiser, 1992].

### 3.1.2 Average Squared Jump Distance

Another performance measure is the average squared jump distance (ASJD)

$$\text{ASDJ}_k^{(r)} = \frac{1}{N} \sum_{i=1}^N |X_{i,k}^{(r)} - X_{i-1,k}^{(r)}|^2$$

for the  $k$ -th component and the  $r$ -th repetition of the Markov chain [Lau and Krumscheid, 2019]. Larger ASJD values are preferred, as they indicate that the state-space is better explored [Lau and Krumscheid, 2019].

## 3.2 Experiments

### 3.2.1 Target Distributions

As experiments to test the performance of the adaptive plateau-based MCMC method [Lau and Krumscheid, 2019] examines four target distributions with the Markov chain. The target distributions are

- $\pi_1$  : a mixture of 4-dimensional Gaussians  $\frac{1}{2}\mathcal{N}(\mu_1, \Sigma_1) + \frac{1}{2}\mathcal{N}(\mu_2, \Sigma_2)$  with  $\mu_1 = (5, 5, 0, 0)^T$ ,  $\mu_2 = (15, 15, 0, 0)^T$ ,  $\Sigma_1 = \text{diag}(6.25, 6.25, 6.25, 0.01)$ ,  $\Sigma_2 = \text{diag}(6.25, 6.25, 0.25, 0.01)$ ,
- $\pi_2$  : an 8-dimensional banana distribution with density  $f \circ \phi$ , where  $f$  is the density of an 8-dimensional normal distribution  $\mathcal{N}(0, \Sigma_3)$  with  $\Sigma_3 = \text{diag}(100, 1, \dots, 1)$  and  $\phi(x) = (x_1, x_2 + 0.03x_1^2 - 3, x_3, \dots, x_8)$  for  $x \in \mathbb{R}^8$ ,
- $\pi_3$  : a perturbed 2-dimensional gaussian with unnormalized probability density  $\tilde{\pi}_3(x) = \exp(-x^T A x - \cos(\frac{x_1}{0.1}) - 0.5 \cos(\frac{x_2}{0.1}))$  for  $x \in \mathbb{R}^2$  with  $A = \begin{pmatrix} 1 & 1 \\ 1 & 3/2 \end{pmatrix}$ , and
- $\pi_4$  : a 1-dimensional bi-stable distribution with unnormalized distribution  $\tilde{\pi}_4(x) = \exp(-x^4 + 5x^2 - \cos(\frac{x}{0.02}))$  for  $x \in \mathbb{R}$ .

These distributions are shown in Figure 2, which is based on [Lau and Krumscheid, 2019, Figure 3]. The representation of  $\pi_2$  in Figure 2b results from a dimensionally reduced simplification, since a marginal with 6 integration variables was computationally too expensive. One can also recognize this by the fact that the probability density here is greater than that in [Lau and Krumscheid, 2019].

### 3.2.2 Simulation Parameter Settings

For each of the 4 target distributions  $R = 200$  MCMC simulations are performed with  $N = 4000$  iterations for  $\pi_1$ ,  $N = 10000$  iterations for  $\pi_2$ , and  $N = 3000$  iterations for  $\pi_3$  and  $\pi_4$ . The remaining parameters are the same for all simulations. These are

- $M = 5$  different plateau proposal distributions,
- $\delta = \delta_1 = 2$  widths of plateaus,
- $\sigma = 0.05$  and  $\sigma_0 = \sigma_1 = 3$  flatness of the plateau tails,
- $\eta_1 = \eta_2 = 0.4$  minimum amount for an adaptation to occur,
- $L = 40$  length of the interval between adaptations, and
- a burn-in portion of 0.5 of all iterations.

These parameters are set exactly as in [Lau and Krumscheid, 2019] except  $L$ , for which there is no exact specification.

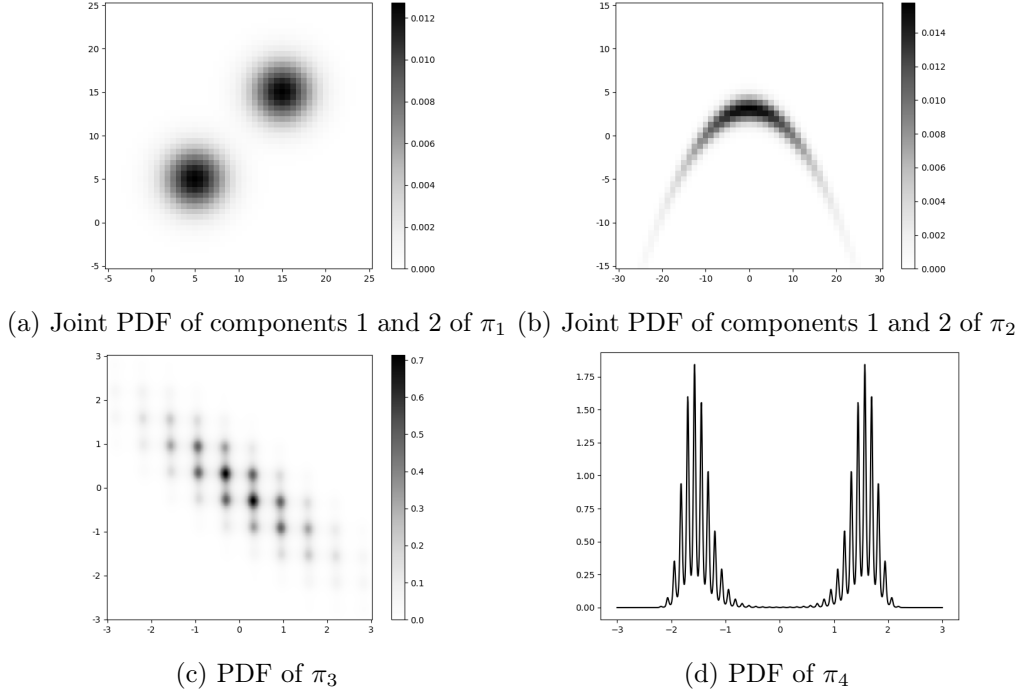


Figure 2: Overview of target distributions

### 3.3 Results

The violin plots belonging to the experiments can be found for the target distributions  $\pi_1$ ,  $\pi_2$ ,  $\pi_3$  and  $\pi_4$  in the figures 3, 4, 5 and 6. One finds the component-wise representation of the ACT distribution in the subfigures 3a, 4a, 5a and 3a, and the component-wise representation of the ASJD distribution in the subfigures 3b, 4b, 5b and 6b. In order to better estimate the results, the tables 1, 2 and 3 contain the median, mean, minimum and maximum in rounded form listed for ACT, log-ACT and ASJD results of the experiments.

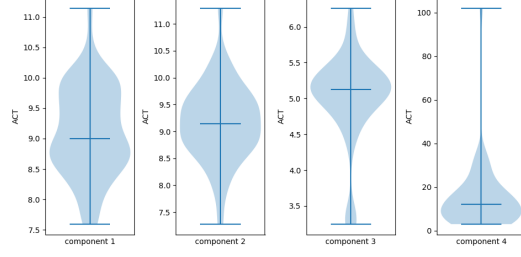


|        |                                |  |
|--------|--------------------------------|--|
|        | $\pi_1$                        | $\pi_2$  |
| median | 8.999, 9.149, 5.126, 12.131    | 82.767, 88.027, 3.179, 3.17, 3.173, 3.168, 3.17, 3.181     |
| mean   | 9.117, 9.171, 5.019, 14.828    | 92.596, 102.911, 3.168, 3.159, 3.164, 3.156, 3.165, 3.167  |
| min    | 7.594, 7.277, 3.247, 3.042     | 54.33, 46.736, 3.022, 3.007, 3.019, 3.034, 3.032, 3.03     |
| max    | 11.145, 11.295, 6.262, 102.043 | 231.547, 246.187, 3.354, 3.318, 3.325, 3.323, 3.347, 3.354 |
|        | $\pi_3$                        | $\pi_4$  |
| median | 7.769, 8.155                   | 3.62   |
| mean   | 7.821, 8.288                   | 3.635  |
| min    | 6.569, 6.84                    | 3.337  |
| max    | 9.722, 10.042                  | 4.36   |

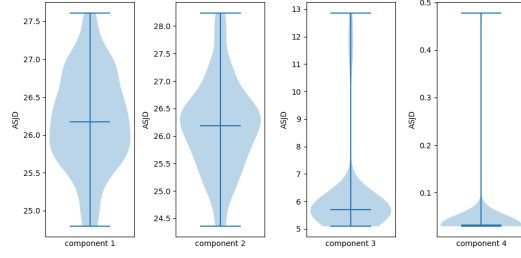
Table 1: Rounded statistical results of ACT component-wise

|        |                            |  |
|--------|----------------------------|--|
|        | $\pi_1$                    | $\pi_2$  |
| median | 2.197, 2.214, 1.634, 2.496 | 4.416, 4.478, 1.157, 1.154, 1.155, 1.153, 1.154, 1.157 |
| mean   | 2.207, 2.213, 1.605, 2.382 | 4.488, 4.551, 1.153, 1.15, 1.152, 1.149, 1.152, 1.152  |
| min    | 2.027, 1.985, 1.178, 1.113 | 3.995, 3.845, 1.106, 1.101, 1.105, 1.11, 1.109, 1.108  |
| max    | 2.411, 2.424, 1.835, 4.625 | 5.445, 5.506, 1.21, 1.199, 1.202, 1.201, 1.208, 1.21   |
|        | $\pi_3$                    | $\pi_4$  |
| median | 2.05, 2.099                | 1.286  |
| mean   | 2.053, 2.111               | 1.289  |
| min    | 1.882, 1.923               | 1.205  |
| max    | 2.274, 2.307               | 1.473  |

Table 2: Rounded statistical results of log-ACT component-wise



(a) Component-wise ACT distributions for target distribution  $\pi_1$



(b) Component-wise ASJD distributions for target distribution  $\pi_1$

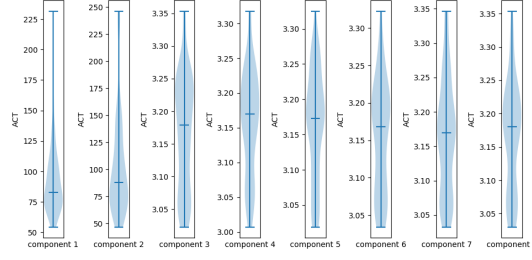
Figure 3: Violin plots of performance measures for target distribution  $\pi_1$

|        | $\pi_1$                       | $\pi_2$   |
|--------|-------------------------------|---|
| median | 26.172, 26.186, 5.691, 0.032  | 9.508, 2.907, 2.948, 2.945, 2.942, 2.938, 2.953, 2.952  |
| mean   | 26.191, 26.169, 6.158, 0.041  | 9.791, 2.922, 2.972, 2.965, 2.949, 2.96, 3.033, 2.977   |
| min    | 24.792, 24.357, 5.095, 0.029  | 8.315, 2.837, 2.852, 2.858, 2.854, 2.841, 2.863, 2.86   |
| max    | 27.612, 28.239, 12.865, 0.478 | 21.963, 3.306, 3.698, 3.522, 3.161, 3.534, 3.651, 3.648 |
|        | $\pi_3$                       | $\pi_4$   |
| median | 1.641, 0.894                  | 3.527   |
| mean   | 1.65, 0.905                   | 3.514   |
| min    | 1.512, 0.856                  | 2.641   |
| max    | 1.779, 1.166                  | 3.736   |

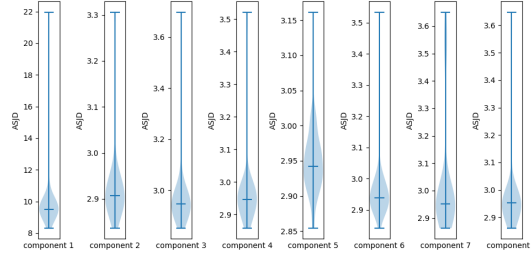
Table 3: Rounded statistical results of ASJD component-wise

## 4 Discussion

The results partly differ significantly from those of the simulations in [Lau and Krumscheid, 2019]. However, the ratios of the order of magnitude between the components of the ACT and

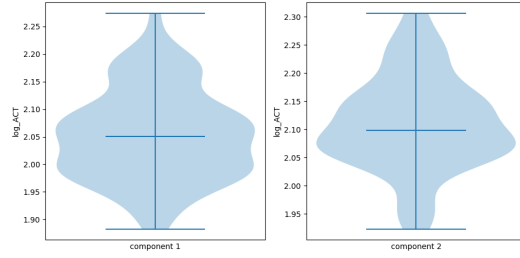


(a) Component-wise ACT distributions for target distribution  $\pi_2$

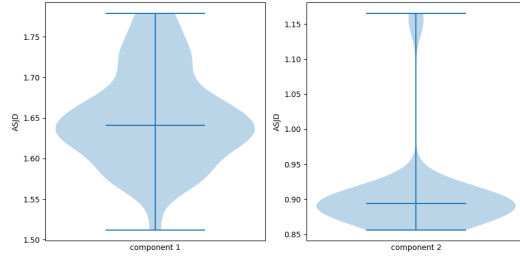


(b) Component-wise ASJD distributions for target distribution  $\pi_2$

Figure 4: Violin plots of performance measures for target distribution  $\pi_2$



(a) Component-wise ACT distributions for target distribution  $\pi_3$



(b) Component-wise ASJD distributions for target distribution  $\pi_3$

Figure 5: Violin plots of performance measures for target distribution  $\pi_3$

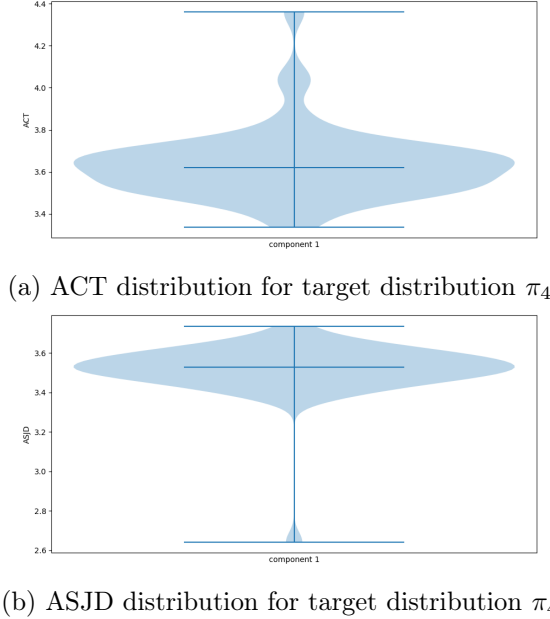


Figure 6: Violin plots of performance measures for target distribution  $\pi_4$

ASJD values appear to be largely consistent. In the ACT distributions, there is a close similarity between [Lau and Krumscheid, 2019] and this reimplement (tables ??) in the target distributions  $\pi_2$  and  $\pi_4$ , but large deviations for  $\pi_1$  and  $\pi_3$ . There is also a high similarity in the ASJD distributions for the target distributions  $\pi_2$  and  $\pi_4$  between [Lau and Krumscheid, 2019] and this project (Table 3). For the target distributions  $\pi_1$  and  $\pi_3$  we can observe differences in the values, but the ratios between the components are very similar to the ones in [Lau and Krumscheid, 2019]. Additionally I had the subjective impression that there were big variances in the ACT and ASJD distributions in experiments for the target distribution  $\pi_4$ . Unfortunately I could not compare to the MCMC algorithms MH, AG1 and AG2 [Lau and Krumscheid, 2019] due to time constraints. Therefore the higher effectiveness of the adaptive plateau-based multiple-try MCMC algorithm, which the authors have concluded in [Lau and Krumscheid, 2019], can I neither confirm nor dispute.

## 5 Technical Details of Implementation

The reimplement of [Lau and Krumscheid, 2019] is written in Python 3 and uses the packages Numpy, Scipy and Matplotlib. The four simulations for reproducing the

MCMC results can be started with the script `run_simulations.py`, which runs all four simulations in parallel using `multiprocessing`. The adjustable parameters can be found under `adjustable parameters` in the script. They follow the naming convention from this report and the paper [Lau and Krumscheid, 2019]. After the simulations have been run, the script `create_violin_plots.py` can be used to create the violin plots for the results. Furthermore there are two scripts `create_fig_2b.py` and `create_fig_3.py` with which reproductions of the Figures 2b and 3 from [Lau and Krumscheid, 2019] can be produced. If you want to use the MCMC runs already provided in the cloud, one has to copy the extracted Numpy-arrays into the folder `simulations`.

## References

- [Foreman-Mackey, 2020] Foreman-Mackey, D. (2020). Autocorrelation time estimation by dan foreman-mackey. <https://dfm.io/posts/autocorr/>. Accessed: 2020-03-20.
- [Lau and Krumscheid, 2019] Lau, F. D.-H. and Krumscheid, S. (2019). Plateau proposal distributions for adaptive component-wise multiple-try metropolis.
- [Liu et al., 2000] Liu, J., Liang, F., and Wong, W. (2000). The multiple-try method and local optimization in metropolis sampling. *Journal of The American Statistical Association - J AMER STATIST ASSN*, 95:121–134.
- [Metropolis et al., 1953] Metropolis, Nicholas, W., A., Rosenbluth, N., M., H., A., Teller, and Edward (1953). Equation of state calculations for fast computing machines. *Journal of Chemical Physics* 6, 21:1087–.
- [Peng, 2018] Peng, R. D. (2018). Advanced statistical computing. <https://bookdown.org/rdpeng/advstatcomp/rejection-sampling.html>. Accessed: 2020-03-17.
- [Schmeiser, 1992] Schmeiser, B. (1992). Practical markov chain monte carlo. *Statistical Science*, 7.
- [Yang et al., 2019] Yang, J., Levi, E., Craiu, R. V., and Rosenthal, J. S. (2019). Adaptive component-wise multiple-try metropolis sampling. *Journal of Computational and Graphical Statistics*, 28(2):276–289.