Irreversible Parallel Tempering and an application to a Bayesian nonparametric latent feature model

Fan Wu

Somerville College

University of Oxford

A dissertation submitted in partial fulfilment of the requirements for the degree of

Master of Science in Applied Statistics

Trinity 2017

Abstract

In this dissertation we address the diffusive random walk behavior exhibited by the replicas in parallel tempering and propose novel non-reversible versions of parallel tempering using lifting ideas in order to suppress this behavior, increase the round trip rate and accelerate the discovery of new modes. We introduce two versions of the lifted parallel tempering algorithm and extend them to the case where multiple swaps are attempted in each iteration. In order to experimentally characterize situations when the non-reversible algorithms have better performance than parallel tempering, we test the algorithms on a Gaussian mixture model, a default multimodal target distribution which was also considered in [22]. We find that the non-reversible algorithms compare favorably with standard parallel tempering if the number of temperatures is large relative to the number of iterations the algorithms are run for. Finally, we apply a non-reversible parallel tempering algorithm to a bayesian nonparametric latent feature model and demonstrate a situation where the Gibbs sampler, which was introduced for this model by Griffiths and Ghahramani in [13], fails to recover the underlying features, while parallel tempering is able to do so.

First and foremost, I would like to express my sincere gratitude to my academic supervisor, Professor Arnaud Doucet, for his helpful guidance, suggestions and encouragement throughout this dissertation. I would also like to thank Professor Alexandre Bouchard-Côté for his insightful comments and advice.

Contents

1. Introduction

Markov chain Monte Carlo (MCMC) methods are widely used to approximate high dimensional integrals and are a commonly used tool in Bayesian statistics. Popular MCMC algorithms include the Metropolis-Hastings algorithm, which was introduced by Metropolis et al. in [25] and is an effective tool to sample from a distribution given that we can evaluate its density up to proportionality.

However, when the target distribution has multiple well separated modes, the Metropolis-Hastings algorithm struggles to traverse the different modes and tends to get stuck in one of them. Parallel tempering, or replica exchange, was devised by Swendson and Wang in 1986 [37] and has later been formulated in a more familiar way by Geyer in 1991 [9]. It has since advanced to one of the go-to methods for sampling from multimodal target distributions in a wide range of application areas including physics, chemistry and biology.

In parallel tempering, N Markov chains are simulated in parallel at different temperatures. The idea is that chains at high temperatures can explore large areas of the state space, while at low temperatures precise local samples from a part of the state space can be obtained. By allowing the Markov chains to exchange their configurations occasionally, the low temperature chains can access regions of the state space discovered by the high temperature chains they would otherwise have not visited in a reasonable amount of computational time. Coupling the Markov chains like this implies that each individual chain is not Markov anymore, however the cartesian product of all N chains is a Markov chain with the cartesian product of the individual invariant distributions π_n as invariant distribution.

A widely used exchange scheme is to randomly choose two adjacent temperatures and swap these replicas with some probability given by the Metropolis ratio, that is these swap moves are performed within the Metropolis-Hastings framework, satisfy detailed balance and generate a reversible Markov chain (see e.g. [6]). However, the trace of one particular replica in the temperature space behaves like a lazy random walk: the replica moves up or down randomly, or it does not move at all. It is noted in [5] that, in order to travel distance n, a random walk requires time of order n^2. Since we want replicas to move up and down the temperature space fast, that is obtain a high round trip rate (which was considered as measure of performance in e.g. [15] and [24]), this random walk behaviour is undesirable.

Many ideas have been proposed in order to improve parallel tempering. The number of and spacing between temperatures has been studied in many works including [1], [20], [21], [31] and [33], and a general consensus is that the acceptance probability between two adjacent temperatures should be equal for all temperatures and that an acceptance probability of roughly $20.0 - 23.4\%$ and a geometric progression for the temperatures, that is $\beta_i = q^{N-i}$ for $i = 1, ..., N$ and some $q \in (0, 1)$, are generally advisable. Other schemes for choosing an optimal temperature set have been proposed. For instance, an adaptive parallel tempering algorithm which continuously updates the temperature parameters along the run has been proposed by Miasojedow et al. in [26]. Katzgraber et al. proposed a feedback algorithm in [18], where the temperature set is iteratively updated in order to maximize the round trip rate.

Other enhancements for parallel tempering have been studied which target the exchange scheme, under which the swaps are attempted. Lingenheil et al. compared four different schemes measured by their round trip rates in [24] and found that the deterministic even/odd algorithm, which alternatingly attempts to swap all even pairs of temperatures $(2i, 2i + 1)$ and odd pairs $(2i - 1, 2i)$, is the best exchange scheme of the ones considered. In [17], Itoh and Okumura consider permutations of all replicas instead of only exchanging two adjacent replicas. This is realized using the Suwa-Todo algorithm, which was introduced in [36] and achieves a reduction of the rejection rate in MCMC algorithms. We could realize permutations by performing multiple swaps of adjacent replicas, and if we allow each replica to be in more than one pair of temperatures, then a replica can also move by more than one temperature in one swap move. While all these extensions aim at improving mixing and the round trip rate in parallel tempering, the issue of the diffusive random walk behaviour of each replica in the temperature space is not addressed.

Non-reversible MCMC algorithms, which violate the sufficient but not necessary detailed balance condition, have sparked a lot of interest and were shown to often perform better than reversible algorithms (see e.g. [2, 3, 5, 7, 36, 43]). However, non-reversible algorithms are often not easy to design. The lifting idea is a popular approach to design non-reversible algorithms on a discrete state space, see e.g. [4, 28, 45]. The idea is to enlarge the state space with a lifting variable, breaking reversibility and introducing persistency which allows better exploration of the state space.

Inspired by this lifting idea we propose non-reversible parallel tempering algorithms within the Generalized Metropolis-Hastings framework introduced in [23], which are intended to suppress the diffusive random walk behaviour in the temperature space. In [5] it is shown that such a non-reversible algorithm achieves a squareroot reduction of the mixing time compared to a random walk, thus suggesting that lifting is a sensible approach to this end. We introduce a lifting variable (n, ϵ), indicating which replica should be swapped and with which neighbour this swap should be performed in the next iteration. This variable keeps the same direction until it hits the highest or lowest temperature or a swap is rejected. This way, replicas are "pushed" in the same direction in order to increase the round trip rate. We further show how this algorithm can be extended to multiple copies of the lifting variable without deteriorating the persistency introduced by a single lifting variable.

We compare these new methods with standard parallel tempering in a Gaussian mixture model, which was also considered in [22] and has a multimodal posterior distribution, as each permutation of the mixture components constitues a mode. In this setting we experimentally characterize situations in which the new methods show a better performance in terms of the round trip rate and also the number of modes that are discovered at the lowest temperature.

Finally, we apply the new parallel tempering algorithm to a Bayesian nonparametric latent feature model. With the development of the Indian Buffet Process [12], a prior for binary matrices with a finite number of rows and infinitely many columns was introduced, which can be used to express the latent feature structure underlying the given data. Given that the posterior distribution tends to be highly multimodal, classical inference algorithms such as the Gibbs sampler from [13] and [14], an accelerated version of the Gibbs sampler proposed in [44] and the slice sampler proposed in [38], which is based on the Stick-breaking construction of the Indian Buffet Process, often struggle to find the global optimum and recover the underlying feature structure. It therefore seems natural to apply parallel tempering.

Structure of this dissertation

In section 2 an overview of the parallel tempering algorithm and the Generalized Metropolis-Hastings, on which the following algorithms are based, is given. We propose non-reversible algorithms inspired by the lifting idea presented in [27] in section 3 and extend them to the case where multiple swaps are attempted in each iteration. In section 4 we compare the new non-reversible algorithms with standard parallel tempering in a Gassian mixture model. We introduce the bayesian nonparametric latent feature model and the commonly used Indian Buffet Process prior in section 5 and apply parallel tempering to it. Finally, we summarize our findings and discuss limitations and possible further developments in section 6.

2. Background

In this section we briefly present the parallel tempering algorithm, which was introduced in [9], and the Generalized Metropolis-Hastings framework described in [23], on which the following irreversible algorithms will be based.

2.1. Parallel Tempering

Let $(\Omega, \mathcal{A})$ be a measurable space with a probability measure $\pi(dx)$ defined on it, from which we are interested in obtaining samples. When π is multimodal, sampling methods such as the Metropolis-Hastings algorithm often perform poorly. The idea of parallel tempering is to run N Markov chains $(X_t^{(n)})_{t\geq 0}$ $(n = 1, ..., N)$ with different target distributions π_n in parallel. These distributions should be such that $\pi_N = \pi$ is the original distribution, and π_n are related distributions, for which it is easier to traverse the different modes. Occasionally a swap move is proposed, where two chains i and j are chosen at random their states $x^{(i)}$ and $x^{(j)}$ are swapped with probability

$$\frac{\pi_i(x^{(j)})\pi_j(x^{(i)})}{\pi_i(x^{(i)})\pi_j(x^{(j)})}.$$

This move preserves detailed balance, so the Markov chain $(X_t)_{t\geq 0} = (X_t^{(1)}, ..., X_t^{(N)})_{t\geq 0}$ has invariant distribution $\pi_1 \times \pi_2 \times ... \times \pi_N$ on the product space Ω^N.

We introduce some notation that will be used throughout this dissertation.

Let $0 < \beta_1 < \beta_2 < ... < \beta_N = 1$ be the inverse temperatures, then we write π_n for the probability measure which is defined by $\pi_n(dx) \propto \pi(dx)^{\beta_n}$. This is a common choice for the auxiliary distributions π_n.

We write $\tilde{\pi} = \pi_1 \times \pi_2 \times ... \times \pi_N$ for the product measure on Ω^N.

For an element in the product space $x = (x^{(1)}, ..., x^{(N)}) \in \Omega^N$ we will write
$x_{(i,j)} = (x^{(1)}, ..., x^{(i-1)}, x^{(j)}, x^{(i+1)}, ...x^{(j-1)}, x^{(i)}, ..., x^{(N)})$, that is the element with the states i and j swapped.

We refer to $(X_t^{(n)})_{t\geq 0}$ as the n^{th} chain, and by the n^{th} replica we mean the sequence of states $(x_0^{(n)}, x_1^{(n)}, ..., x_t^{(n)}, x_{t+1}^{(m)}, x_{t+2}^{(m)}, ...)$, where a successful swap between chains n and m took place at time t. That is, the n^{th} chain always moves at temperature β_n, whereas we

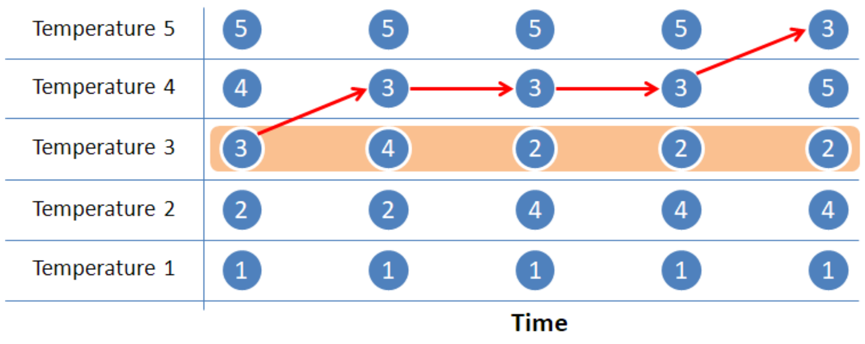

Figure 2.1.: Example with 5 temperatures. Chain 3 in is marked with the orange box, replica 3 moves along the red arrows

follow the n^{th} replica in the temperature spaces and assign a new temperature to it after successful swaps.

Taking π to a power close to zero flattens the distribution, so the Markov chain $(X_t^{(1)})_{t\geq 0}$ will traverse the modes much more easily than $(X_t^{(N)})_{t\geq 0}$, which is the chain with invariant distribution $\pi_N = \pi$ we are interested in. Using the swap moves the modes discovered by $(X_t^{(1)})_{t\geq 0}$ can be communicated up to the chain we are interested in.

It is well known that it is advisable to only propose swapping two adjacent replicas (see e.g [6]), as this way no highly unlikely swaps between two replicas at temperatures far apart from each other are proposed. In the following we will compare our non-reversible algorithms with the following parallel tempering algorithm, where we sample two adjacent replicas at random for the swap attempts.

Algorithm 1 Parallel Tempering - Adjacent

```
1: Input: distribution $\pi$, temperatures $0 < \beta_1 < ... < \beta_N = 1$, number of swap attempts
   $K$, number of MH steps between two swap attempts $s$, initial state $start$

2: sample $s$ MH steps with initial states $start$
3: for $k = 2$ to $K$ do
4:     sample $i$ from $\{1, ..., N-1\}$, $U \sim$ Unif(0,1)
5:     compute $\alpha = \min(1, \frac{\pi_i(x^{(i+1)}_{(k-1)s})\pi_{i+1}(x^{(i)}_{(k-1)s})}{\pi_i(x^{(i)}_{(k-1)s})\pi_{i+1}(x^{(i+1)}_{(k-1)s})})$
6:     if $U < \alpha$ then swap the states $x^{(i)}_{(k-1)s}$ and $x^{(i+1)}_{(k-1)s}$
7:     end if
8:     sample $s$ MH steps with initial states $x_{(k-1)s}$
9: end for
```

Finally, it is noted in [5] that a random walk requires steps of order n^2 in order to travel distance n. In our case the replicas do not perform a symmetric random walk; rather, the probabilities to move up and down depend on the current configuration and are random, and a replica can also not move at all. Nonetheless, this provides us with an intuition that when the number of temperatures becomes large, the number of swap attempts needs to grow roughly quadratically in the number of temperatures in order to ensure that the replicas can still move up and down the temperature space.

2.2. Generalized Metropolis-Hastings Framework

The Generalized Metropolis-Hastings framework can be used to construct non-reversible Markov chain Monte Carlo algorithms and was introduced in [23].

As before assume we are interested in sampling from a distribution $\pi(dx)$ on a measurable space $(\Omega, \mathcal{A})$. As in the Metropolis-Hastings algorithm we need a proposal kernel $Q : (\Omega, \mathcal{A}) \to [0, 1]$. In the GMH framework we additionally need a measure preserving involution:

Assumption 2.2.1. *Let $\pi(dx)$ be a probability distribution on the measurable space $(\Omega, \mathcal{A})$, $S : \Omega \to \Omega$ be an involution, that is $S = S^{-1}$, and the proposal kernel $Q : (\Omega, \mathcal{A}) \to [0, 1]$ be a Markov kernel. Assume the following holds:*

- *The involution is measure preserving, that is*
$$\pi(S(dx)) = \pi(dx).$$
- *The Metropolis-Hastings ratio*
$$r_{GMH}(x, x') = \frac{\pi(S(dx'))Q(S(x'), S(dx))}{\pi(dx)Q(x, dx')}$$
exists and is positive for almost all $(x, x') \in \Omega^2$ with respect to the measure $\pi(dx)Q(x, dx')$.

Under these assumptions the Generalized Metropolis-Hastings algorithm proceeds as follows:

Algorithm 2 Generalized Metropolis-Hastings Algorithm

```
1: Input: distribution $\pi$, initial state $x_0$, proposal kernel $Q$, involution $S$

2: for $t = 1$ to $T$ do
3:     sample $x' \sim Q(x_{t-1}, \cdot)$, $U \sim$ Unif(0,1)
4:     compute $\alpha_{GMH}(x_{t-1}, x') = \min(1, \frac{\pi(S(dx'))Q(S(x'),S(dx_{t-1}))}{\pi(dx_{t-1})Q(x_{t-1},dx')})$
5:     if $U < \alpha_{GMH}(x_{t-1}, x')$ then set $x_t = x'$
6:     else set $x_t = S(x_{t-1})$
7:     end if
8: end for
```

This algorithm corresponds to the transition kernel

$$K_{GMH}(x,dx') = \alpha_{GMH}(x,x')Q(x,dx') + \Big(1 - \int \alpha_{GMH}(x,y)Q(x,dy)\Big)\delta_{S(x)}(dx'), \qquad (2.1)$$

that is unlike in the traditional Metropolis-Hastings algorithm an involution S is applied upon rejection.

The transition kernel K_{GMH} satisfies detailed balance up to transformation S,

$$\pi(dx)K_{GMH}(x,dx') = \pi(dx')K_{GMH}(S(x'),S(dx)), \qquad (2.2)$$

which implies that the Markov chain with transition kernel K_{GMH} has invariant distribution π, see [23].

All following algorithms will be in this framework and have 2.1 as transition kernel. If π is the distribution from which we intend to obtain samples it is generally not easy to find a measure preserving involution S (which is not the identity). Instead we embed Ω into a larger space, say $\Omega \times \mathcal{X}$, and introduce an auxiliary variable $v \in \mathcal{X}$. We then consider a distribution of the form $\rho(dx,dv) = \pi(dx)\psi(dv)$, and since we get to choose ψ it is straightforward to find a measure preserving involution S of the form $S(x,v) = (x,v')$, that is S only modifies the auxiliary variable.

If we consider a deterministic proposal of the form

$$Q((x,v),(dx',dv')) = \delta_{\phi(x,v)}(dx',dv'), \qquad (2.3)$$

then, in order to satisfy Assumption 2.2.1, we need to ensure that the measures appearing in the Metropolis-Hastings ratio,

$$\rho(dx,dv)Q((x,v),(dx',dv')) = \rho(dx,dv)\delta_{\phi(x,v)}(dx',dv') \qquad (2.4)$$

and

$$\rho(S(dx',dv'))Q(S(x',v'),S(dx,dv)) = \rho(S(dx',dv'))\delta_{\phi(S(x',v'))}(S(dx,dv)) \qquad (2.5)$$

are non-singular. 2.4 being non-singular implies

$$(x',v') = \phi(x,v),$$

and 2.5 is non-singular if

$$\begin{aligned} S(x,v) &= \phi(S(x',v')) \\ \Leftrightarrow (x',v') &= S \circ \phi^{-1} \circ S(x,v). \end{aligned}$$

Together this implies

$$\begin{aligned} \phi(x,v) = (x',v') &= S \circ \phi^{-1} \circ S(x,v) \\ \Leftrightarrow \qquad \phi^{-1} &= S \circ \phi \circ S, \end{aligned}$$

and the acceptance probability simplifies to

$$\alpha_{GMH}((x,v),(x',v')) = \min\Big(1, \frac{\rho(\phi(dx',dv'))}{\rho(dx,dv)}\Big).$$

3. Irreversible Parallel Tempering Algorithms

In this section we will present non-reversible parallel tempering algorithms based on the GMH framework described above. For the moves of each individual chain we will use the Metropolis-Hastings algorithm, and we will only discuss how the swaps are performed. The transition kernels we consider will be of the form 2.1, and we will consider deterministic proposals as in 2.3. That is, we need to specify the extended state space $\Omega \times \mathcal{X}$, the extended target distribution $\rho(dx, dv) = \pi(dx)\psi(dv)$, the proposal kernel Q, which is defined by the function ϕ, and the involution S. In order to ensure that assumption 2.2.1 is satisfied we check that S is a measure preserving involution and that $\phi^{-1} = S \circ \phi \circ S$ holds. These computations are given in the appendix.

3.1. Lifted Parallel Tempering

In standard parallel tempering each replica exhibits a diffusive random walk behavior in the temperature space $\{1, ..., N\}$. However, it would be desirable if each replica moved up and down the temperature space in a more persistent way than that, thus increasing the round trip rate, which we will define in section 4.2.

We therefore propose a lifted version of parallel tempering, where we "push" a replica in the same direction until a swap is rejected. We introduce a lifting variable (n_t, ϵ_t), indicating the current state n_t, from which the next swap will be attempted, and the direction $\epsilon_t \in \{-1, 1\}$, in which to swap, that is we propose to swap the states $x_t^{(n_t)}$ and $x_t^{(n_t+\epsilon_t)}$. If this swap is accepted we set $n_{t+1} = n_t + \epsilon_t$ and $\epsilon_{t+1} = \epsilon_t$ unless we hit one of the "border" states 1 or N, i.e. $n_{t+1} \in \{1, N\}$, in which case we would set $\epsilon_{t+1} = -\epsilon_t$. If the swap is rejected we set $n_{t+1} = n_t$ and $\epsilon_{t+1} = -\epsilon_t$, again unless $n_{t+1} \in \{1, N\}$, in which case we would keep $\epsilon_{t+1} = \epsilon_t$. That is, we keep swapping in the same direction ϵ_t until we reach the state 1 or N or reject a proposed swap, when we would reverse the direction and replace it with $-\epsilon$. We also note that in this algorithm the only the replica, in which the first current state is, can move in the temperature space; all other replica can only visit their neighbours. The extended state space is $\Omega^N \times \Big(\{1, 2, ..., N\} \times \{-1, 1\} \setminus (\{1\} \times \{-1\} \cup \{N\} \times \{1\})\Big)$.

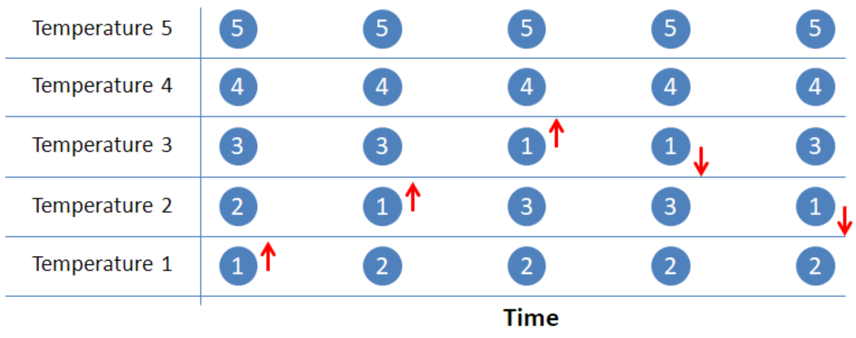

Figure 3.1.: Example with 5 temperatures. Red arrows indicate the current state and the direction of the next swap. The third swap was rejected.

The extended target distribution is $\tilde{\pi} \times \text{Unif}\{1, ..., N\} \times \text{Unif}\{-1, 1\}$.
The transition kernel for the swap in this and all following algorithms is given by

$$K_{LPT}(z, dz') = \alpha_{LPT}(z, z')Q(z, dz') + \left(1 - \int \alpha_{LPT}(z, w)Q(z, dw)\right)\delta_{S(z)}(dz')$$

where $z = (x, n, \epsilon)$ is an element in the extended state space.

Algorithm 3 Lifted Parallel Tempering

```
1: Input: distribution $\pi$, temperatures $0 < \beta_1 < ... < \beta_N = 1$, number of swap attempts
   $K$, number of MH steps between two swap attempts $s$, initial state *start*

2: set $n = 1$, $\epsilon = 1$
3: sample $s$ MH steps with initial states *start*
4: for $k = 2$ to $K$ do
5:     sample $U \sim$ Unif(0,1)
6:     compute $\alpha = \min(1, \frac{\pi_n(x^{(n+\epsilon)}_{(k-1)s})\pi_{n+\epsilon}(x^{(n)}_{(k-1)s})}{\pi_n(x^{(n)}_{(k-1)s})\pi_{n+\epsilon}(x^{(n+\epsilon)}_{(k-1)s})})$
7:     if $U < \alpha$ then swap the states $x^{(n)}_{(k-1)s}$ and $x^{(n+\epsilon)}_{(k-1)s}$
8:                set $n = n + \epsilon$, $\epsilon = \epsilon$
9:     else set $n = n$, $\epsilon = -\epsilon$
10:    end if
11:    if $n = 1$ then set $\epsilon = 1$
12:    end if
13:    if $n = N$ then set $\epsilon = -1$
14:    end if
15:    sample $s$ MH steps with initial states $x_{(k-1)s}$
16: end for
```

The acceptance probability is given by

$$\alpha_{LTP}(z, z') = \min\left\{1, \frac{\tilde{\pi}(x')}{\tilde{\pi}(x)}\right\}.$$

The measure preserving involution S is given by

$$S(x, n, \epsilon) = \begin{cases} (x, n, -\epsilon) & \text{if} \quad 1 < n < N \\ (x, n, \epsilon) & \text{if} \quad n \in \{1, N\}, \end{cases} \tag{3.1}$$

and the proposal is defined by

$$\phi(x, n, \epsilon) = \begin{cases} (x_{(n,n+\epsilon)}, n+\epsilon, \epsilon) & \text{if} \quad 1 < n+\epsilon < N \\ (x_{(n,n+\epsilon)}, n+\epsilon, -\epsilon) & \text{if} \quad n+\epsilon \in \{1, N\}. \end{cases} \tag{3.2}$$

3.2. Lifted Parallel Tempering - Circle

While the lifted parallel tempering algorithm indeed leads to faster discovery of the modes in certain settings as we will see in section 4, the introduction of such a lifting variable will lead to the following behavior. Assuming that all N chains start in one mode, say mode 1, new modes are only discovered at the high temperatures $(\beta_1, \beta_2, ...)$. Now when a new mode, say mode 2, is discovered and transported up to temperature N, this chain is now in mode 2, and the chain $N-1$ is in mode 1. Since the current state is now N and we force the swap of N and $N-1$, the N^{th} chain will be back in mode 1. Thus mode 1 acts like a "baseline mode" for the N^{th} chain and it spends disproportionally much time in this mode.

We thus propose the following algorithm, where we do not force the lifting variable to turn around at 1 and N, but allow it to jump between the highest and lowest temperatures. While we could think of the temperatures as a line in the previous algorithm, we can now think of a circle where the two extreme temperatures are also connected. To do so we move the current state to 1 after the successful swap of $N-1$ and N, that is, if a swap with $(n_t, \epsilon_t) = (N-1, 1)$ is successful, we swap $x_t^{(N-1)}$ and $x_t^{(N)}$ and set $(n_{t+1}, \epsilon_{t+1}) = (1, 1)$. In order to preserve balance we also have to allow the reverse movement, i.e. we have to be able to move the current state from 1 to $N-1$, and this movement also needs to swap the chains N and $N-1$, that is after a successful swap with $(n_t, \epsilon_t) = (1, -1)$ we swap $x_t^{(N-1)}$ and $x_t^{(N)}$ and set $(n_{t+1}, \epsilon_{t+1}) = (N-1, -1)$. Upon rejection we always set $\epsilon_{t+1} = -\epsilon_t$ regardless of the state n_{t+1}. We do not move the current state to 1 after a successful swap of N and 1, as it is very unlikely to accept this move, therefore we move to state 1 after swapping $N-1$ with N and have to introduce this rather "awkward" swap in the reverse direction. We thus also have to remove N from the state space of the lifting variable.

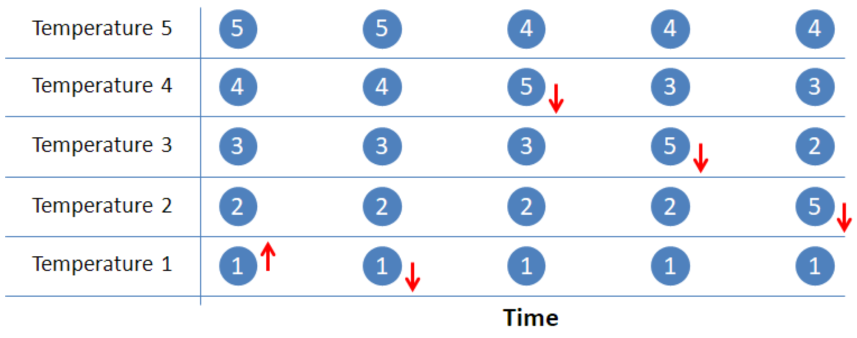

Figure 3.2.: Example with 5 temperatures. Red arrows indicate the current state and the direction of the next swap. The first swap was rejected and the current state jumped from 1 to 4 after the successful second swap.

This way we expect the algorithm to be able to leave the baseline mode, which would take very long to happen in the previous algorithm. This is especially important if we are interested in the relative magnitude of the different modes and want to compare the amount of time chain N spends in the different modes, which would be hardly meaningful if chain N is always forced to return to the mode it started in.

Algorithm 4 Lifted Parallel Tempering - Circle

```
 1: Input: distribution $\pi$, temperatures $0 < \beta_1 < ... < \beta_N = 1$, number of swap attempts
    $K$, number of MH steps between two swap attempts $s$, initial state $start$

 2: set $n = 1$, $\epsilon = 1$
 3: sample $s$ MH steps with initial states $start$
 4: for $k = 2$ to $K$ do
 5:     sample $U \sim$ Unif(0,1), set $n' = n + \epsilon$
 6:     if n'=0 then set $n = N$, $n' = N - 1$
 7:     end if
 8:     compute $\alpha = \min(1, \frac{\pi_n(x^{(n')}_{(k-1)s})\pi_{n'}(x^{(n)}_{(k-1)s})}{\pi_n(x^{(n)}_{(k-1)s})\pi_{n'}(x^{(n')}_{(k-1)s})})$
 9:     if $U < \alpha$ then swap the states $x^{(n)}_{(k-1)s}$ and $x^{(n')}_{(k-1)s}$
10:         if $n' = N$ then set $n' = 1$
11:         end if
12:         set $n = n'$, $\epsilon = \epsilon$
13:     else set $n = n$, $\epsilon = -\epsilon$
14:     end if
15:     sample $s$ MH steps with initial states $x_{(k-1)s}$
16: end for
```

The extended state space now is $\Omega^N \times \{1, ..., N-1\} \times \{-1, 1\}$, that is N is removed as possible current state.

The extended target distribution is $\tilde{\pi} \times \text{Unif}\{1, ..., N-1\} \times \text{Unif}\{-1, 1\}$.

The measure preserving involution S is given by

$$S(x, n, \epsilon) = (x, n, -\epsilon), \tag{3.3}$$

and the proposal is defined by

$$\phi(x, n, \epsilon) = \begin{cases} (x_{(n,n+\epsilon)}, n+\epsilon, \epsilon) & \text{if} \quad (n, \epsilon) \neq (1, -1), (N-1, 1) \\ (x_{(N-1,N)}, 1, 1) & \text{if} \quad (n, \epsilon) = (N-1, 1) \\ (x_{(N-1,N)}, N-1, -1) & \text{if} \quad (n, \epsilon) = (1, -1) \end{cases} \tag{3.4}$$

3.3. Lifted Parallel Tempering - 2 copies

Sindhikara et al. showed in [34] and [35] that proposing swaps moves frequently improves mixing. There are two obvious ways to do so, one being decreasing the number of Metropolis-Hastings steps between two swap proposals, and the other being proposing multiple swaps at once. The former can be done without any modification to the algorithms above, while the latter is straightforward for standard parallel tempering. We can simply sample multiple pairs of temperatures to swap and perform lines $4-7$ for each pair in Algorithm 1. We will show a way to introduce two copies of the lifting variable in the non-reversible parallel tempering algorithms; this can also be applied to the case with more than two copies.

Introducing two copies of the lifting variable aims at speeding up the discovery of new modes at temperature N. New modes are discovered when a new mode is found at a high temperature, and this state is swapped up to temperature N. The idea is that when one lifting variable moves around low temperatures, the second copy can "carry" new modes from high to low temperatures, where this state is passed to the first copy and up to temperature N. Simply introducing two independent copies of the lifting variable will take away some of the persistency we introduced using lifting, since when both copies meet and move in the same direction they would undo each others swaps. Instead we do the following: when the two current states of the two lifting variables are adjacent (i.e. n and $n+1$) and a successful swap between these two states happens, the current states are not changed and the directions are reversed. This way each copy will "turn around" after passing the state to the other copy. Note that this way one lifting variable will always move at higher temperatures than the other, since they can never swap positions. Also, this way only two replica move around the whole temperature space, which is another indication that this implementation is indeed better at preserving the persistency introduced by lifting opposed to, say, two independent copies of the lifting variable, in which case any replica

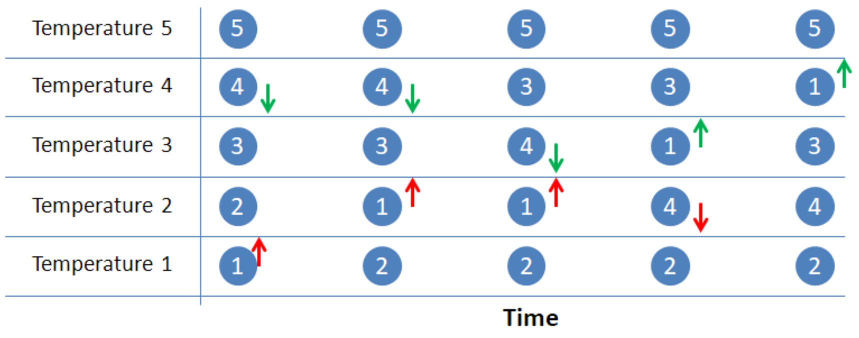

Figure 3.3.: Example with 5 temperatures. Red arrows indicate the current state and the direction the first copy of the lifting variable, green arrows of the second copy. Each step shows only the swap for one copy of the lifting variables, i.e. half a swap attempt.

could be "swapped into" the second lifting variable by the first or vice versa and thus move around the whole temperature space.

The lifting variable now takes the form $(n_1, \epsilon_1, n_2, \epsilon_2)$.
The extended state space is $\Omega^N \times \Big(\{1, ..., N-1\} \times \{-1, 1\} \setminus (\{N\} \times \{1\}) \Big) \times \Big(\{2, ..., N\} \times \{-1, 1\} \setminus (\{1\} \times \{-1\}) \Big)$.

The extended target distribution is $\tilde{\pi} \times \Big(\text{Unif}\{1, ..., N\} \times \text{Unif}\{-1, 1\} \Big)^2$. The state space of the lifting variable takes this form as $1 \leq n_1 < n_2 \leq N$ will always hold in the scheme described above.

For each swap we apply two kernels K_1 and K_2 sequentially, where they operate on the respective copy of the lifting variable.

K_1 is defined as above by the involution S_1 and the proposal kernel Q_1. S_1 is given by

$$S_1(x, n_1, \epsilon_1, n_2, \epsilon_2) = \begin{cases} (x, n_1, -\epsilon_1, n_2, \epsilon_2) & \text{if} \quad n_1 \neq 1 \\ (x, n_1, \epsilon_1, n_2, \epsilon_2) & \text{if} \quad n_1 = 1, \end{cases} \tag{3.5}$$

and the proposal is defined by

$$\phi_1(x, n_1, \epsilon_1, n_2, \epsilon_2) \\ = \begin{cases} x_{(n_1, n_1+\epsilon_1)}, n_1 + \epsilon_1, \epsilon_1, n_2, \epsilon_2) & \text{if} \quad n_1 + \epsilon_1 \notin \{1, n_2\} \\ x_{(n_1, n_1+\epsilon_1)}, n_1 + \epsilon_1, -\epsilon_1, n_2, \epsilon_2) & \text{if} \quad n_1 + \epsilon_1 = 1 \\ x_{(n_1, n_1+\epsilon_1)}, n_1, -\epsilon_1, n_2, -\epsilon_2) & \text{if} \quad n_1 + \epsilon_1 = n_2, n_1 \neq 1 \text{ and } n_2 \neq N \\ x_{(n_1, n_1+\epsilon_1)}, n_1, -\epsilon_1, n_2, \epsilon_2) & \text{if} \quad n_1 + \epsilon_1 = n_2 \text{ and } n_2 = N \\ x_{(n_1, n_1+\epsilon_1)}, n_1, \epsilon_1, n_2, \epsilon_2) & \text{if} \quad n_1 + \epsilon_1 = n_2 \text{ and } n_1 = 1 \end{cases} \tag{3.6}$$

S_2 and ϕ_2 are defined analogously.

Algorithm 5 Lifted Parallel Tempering - 2 copies

```
1: Input: distribution $\pi$, temperatures $0 < \beta_1 < ... < \beta_N = 1$, number of swap attempts $K$, number of MH steps between two swap attempts $s$, initial state start

2: set $n_1 = 1$, $\epsilon_1 = 1$, $n_2 = N$, $\epsilon_2 = -1$
3: sample $s$ MH steps with initial states start
4: for $k = 2$ to $K$ do
5:     for $l = 1$ to $2$ do
6:         sample $U \sim$ Unif(0,1)
7:         compute $\alpha = \min(1, \frac{\pi_{n_l}(x^{(n_l+\epsilon_l)}_{(k-1)s})\pi_{n_l+\epsilon_l}(x^{(n_l)}_{(k-1)s})}{\pi_{n_l}(x^{(n_l)}_{(k-1)s})\pi_{n_l+\epsilon_l}(x^{(n_l+\epsilon_l)}_{(k-1)s})})$
8:         if $U < \alpha$ then swap the states $x^{(n_l)}_{(k-1)s}$ and $x^{(n_l+\epsilon_l)}_{(k-1)s}$
9:
10:            if $n_l + \epsilon_l \neq n_{-l}$ then set $n_l = n_l + \epsilon_l$, $\epsilon_l = \epsilon_l$
11:            else
12:                if $n_l \notin \{1, N\}$ then set $\epsilon_{1,2} = -\epsilon_{1,2}$
13:                end if
14:            end if
15:        else set $n_l = n_l$, $\epsilon_l = -\epsilon_l$
16:        end if
17:        if $n_1 = 1$ then set $\epsilon_1 = 1$
18:        end if
19:        if $n_2 = N$ then set $\epsilon_2 = -1$
20:        end if
21:    end for
22:    sample $s$ MH steps with initial states $x_{(k-1)s}$
23: end for
```

3.4. Lifted Parallel Tempering - 2 copies and Circle

We can also introduce two copies of the lifting variable in algorithm 4.

The extended state space is $\Omega^N \times \Big(\{1, 2, ..., N-1\} \times \{-1, 1\}\Big)^2$.

The extended target distribution is $\tilde{\pi} \times \Big(\text{Unif}\{1, ..., N-1\} \times \text{Unif}\{-1, 1\}\Big)^2$.

For each swap we apply two kernels K_1 and K_2 sequentially, where they operate on the respective copy of the lifting variable.

K_1 is defined as above by S_1

$$S_1(x, n_1, \epsilon_1, n_2, \epsilon_2) = (x, n_1, -\epsilon_1, n_2, \epsilon_2) \tag{3.7}$$

and the proposal is defined by

$$\begin{aligned}&\phi_1(x, n_1, \epsilon_1, n_2, \epsilon_2)\\ &= \begin{cases} x_{(n_1, n_1+\epsilon_1)}, n_1+\epsilon_1, \epsilon_1, n_2, \epsilon_2) & \text{if} \quad n_1+\epsilon_1 \neq n_2 \text{ and } (n_1, \epsilon_1) \neq (1,-1), (N-1, 1) \\ x_{(n_1, n_1+\epsilon_1)}, n_1, -\epsilon_1, n_2, -\epsilon_2) & \text{if} \quad n_1+\epsilon_1 = n_2 \text{ and } (n_1, \epsilon_1) \neq (1,-1), (N-1, 1) \\ x_{(N-1,N)}, 1, 1, n_2, \epsilon_2) & \text{if} (n_1, \epsilon_1) = (N-1, 1) \text{ and } n_2 \neq 1 \\ x_{(N-1,N)}, N, -1, n_2, -\epsilon_2) & \text{if} (n_1, \epsilon_1) = (N-1, 1) \text{ and } n_2 = 1 \\ x_{(N-1,N)}, N-1, -1, n_2, \epsilon_2) & \text{if} (n_1, \epsilon_1) = (1, -1) \text{ and } n_2 \neq N-1 \\ x_{(N-1,N)}, 1, 1, n_2, -\epsilon_2) & \text{if} (n_1, \epsilon_1) = (1, -1) \text{ and } n_2 = N-1 \end{cases}\end{aligned} \tag{3.8}$$

S_2 and Q_2 are defined analogously.

Algorithm 6 Lifted Parallel Tempering - 2 Copies and Circle

```
 1: Input: distribution $\pi$, temperatures $0 < \beta_1 < ... < \beta_N = 1$, number of swap attempts
    $K$, number of MH steps between two swap attempts $s$, initial state *start*

 2: set $n_1 = 1$, $\epsilon_1 = 1$, $n_2 = N$, $\epsilon_2 = -1$
 3: sample $s$ MH steps with initial states *start*
 4: for $k = 2$ to $K$ do
 5:     for $l = 1$ to 2 do
 6:         sample $U \sim$ Unif(0,1), set $n' = n_l + \epsilon_l$
 7:         if n'=0 then set $n_l = N$, $n' = N - 1$
 8:         end if
 9:         compute $\alpha = \min(1, \frac{\pi_{n_l}(x^{(n')}_{(k-1)s})\pi_{n'}(x^{(n_l)}_{(k-1)s})}{\pi_{n_l}(x^{(n_l)}_{(k-1)s})\pi_{n'}(x^{(n')}_{(k-1)s})})$
10:         if $U < \alpha$ then swap the states $x^{(n_l)}_{(k-1)s}$ and $x^{(n')}_{(k-1)s}$
11:             if $n' = N$ then set $n' = 1$
12:             end if
13:             if $n' \neq n_{-l}$ then set $n_l = n'$, $\epsilon_l = \epsilon_l$
14:             else set $\epsilon_{1,2} = -\epsilon_{1,2}$
15:             end if
16:         else set $n_l = n_l$, $\epsilon_l = -\epsilon_l$
17:         end if
18:     end for
19:     sample $s$ MH steps with initial states $x_{(k-1)s}$
20: end for
```

4. Comparison of the Algorithms in a Gaussian Mixture Model

In this chapter we apply the algorithms proposed in chapter 3 to a default multimodal target, which was also considered in [22]. Our goal is to identify situations in which our non-reversible algorithms perform better or worse than the standard algorithms and explore and explain these differences using results obtained from experiments.

4.1. Target distribution

We consider a Gaussian mixture model, which frequently occurs in many statistical problems. Assume that we have k univariate normal mixture components, then our observations $y_1, ..., y_m$ are distributed according to the density

$$p(y_i|\mu_{1:k}, \sigma_{1:k}, w_{1:k}) = \sum_{j=1}^{k} w_j f(y_i|\mu_j, \sigma_j), \tag{4.1}$$

where $f(y_i|\mu_j, \sigma_j^2)$ denotes the univariate normal density with mean μ_i and standard deviation σ_j. w_j is the weight of the k^{th} mixture component, that is $\sum_{j=1}^{k} w_j = 1$ has to hold.

In the following we will consider $k = 4$ mixture components with the means, standard deviations and weights known. We will assume $\mu_{1:4} = (-3, 0, 3, 6)$, $\sigma_j = \sigma$ for some fixed value σ and $w_j = w = \frac{1}{4}$ for $j = 1, ..., 4$.

We sample 100 datapoints $y_1, ..., y_{100}$ from the mixture distribution with density given by 4.1 and want to recover the means of the mixture components $\mu_{1:4}$ assuming that the common standard deviation σ and weights w_j are known, that is our target distribution is the posterior $p(\mu_{1:4}|y_{1:100})$, where we take an uninformative prior such as the uniform distribution on $[-10, 10]^k$. Our target distribution is then given by

$$\begin{aligned}\pi(\mu_{1:4}) = p(\mu_{1:4}|y_{1:100}) &\propto p(y_{1:100}|\mu_{1:4})\mathbf{1}_{[-10,10]^4}(\mu_{1:4}) \\ &= \prod_{i=1}^{100}\sum_{j=1}^{4}\frac{1}{4}f(y_i|\mu_j, \sigma)\mathbf{1}_{[-10,10]^4}(\mu_{1:4}).\end{aligned} \tag{4.2}$$

Since the order of the means μ_j of each component is irrelevant for the resulting mixture distribution, the posterior distribution $p(\mu_{1:4}|y_{1:100})$ is invariant under permutations of $\mu_{1:4}$, that is it has $4! = 24$ symmetric modes.

4.2. Measures of performance

In this section we introduce the ways in which we measure the performance of the different parallel tempering algorithms on the Gaussian mixture model described above.

The first measure we consider is the **number of modes** discovered after each given number of steps. We consider the number of steps rather than the time for simplicity, since all above algorithms have essentially the same time complexity; standard parallel tempering and all non-reversible algorithms proposing the same number of swaps in each iteration need equally many likelihood evaluations, thus they require essentially the same computational time for a given number of steps, as the likelihood evaluations dominate the time complexity in all above algorithms. For each algorithm we count how many of the 24 modes are discovered at temperature N.

Next we consider the **round trip rate**, which was also considered in [15] and [24]. For each replica we count how often this replica visits the highest temperature β_1, runs to the lowest temperature β_N and then back to the highest temperature β_1. One such trip constitutes one round trip. We then compare how many round trips all replicas perform together in a given number of steps for the different algorithms. The round trip rate is the number of round trips divided by the total number of steps.

The round trip rate is closely related to the number of modes discovered, but more focused on the behavior of the replicas in the temperature space, whereas the number of modes discovered also depends on the underlying sampling algorithm being used for each individual replica. Typically a higher round trip rate means that new modes are discovered faster and vice versa. However it is possible that a round trip is performed but a mode which was already discovered is carried to temperature β_N, or that new modes are discovered at temperatures slightly lower than β_1 and communicated to the chain at temperature β_N; thus while these two magnitudes are closly related, there is no one to one correspondence between them. [This paragraph unnecessary?]

When comparing algorithms using different numbers of temperatures it seems unreasonable to run them for the same number of steps, as using more temperatures is computationally more expensive. Therefore we will consider the time, in particular the **time to the discovery of the first new mode** in section 4.6. This measure also has its flaws as it depends on the machine the algorithm is being executed on. If sufficiently many cores are available it is possible to parallelize all replicas, thus not increasing the required computational time compared to an algorithm using fewer temperatures and therefore potentially favoring algorithms utilizing a larger set of temperatures.

Finally we also consider typical convergence diagnostics for MCMC. We investigate the traceplots of the four components of the mean estimate μ and also the autocorrelation times at temperature β_N.

4.3. Experimental setup

We perform the experiment with the following experimental setup. The target distribution is given by equation 4.1 with $w_j = \frac{1}{4}$ for all $j = 1, ..., 4$ and $\mu_{1:4} = (-3, 0, 3, 6)$. Let $\beta_1 < ... < \beta_N = 1$ be our set of inverse temperatures. We consider tempered distributions of the form

$$\begin{aligned}\pi_n(\mu_{1:4}) &\propto \prod_{i=1}^{100} \sum_{j=1}^{4} \frac{1}{4} f(y_i|\mu_j, \sigma)^{\beta_n} \mathbf{1}_{[-10,10]^4}(\mu_{1:4}) \\ &\propto \prod_{i=1}^{100} \sum_{j=1}^{4} \frac{1}{4} f\Big(y_i\Big|\mu_j, \frac{\sigma}{\sqrt{\beta_n}}\Big) \mathbf{1}_{[-10,10]^4}(\mu_{1:4}),\end{aligned} \tag{4.3}$$

that is we take each mixture component rather than the whole distribution π to the power of β_n. This has the simple reason that otherwise $\pi_n(\mu_{1:4})$ would be 0 due to machine precision when it should not be for very high temperatures (i.e. $\beta_n \approx 0$), small standard deviations σ and values $\mu_{1:4}$ far away from the modes.

It was suggested in [20] and [31] to use a geometric progression for the temperatures, that is $\beta_i = q^{N-i}$ for some $q \in (0, 1)$ and $i = 1, ..., N$. This seems to be appropriate also in our case, since

$$d_{TV}(\mathcal{N}(0, \sigma_1), \mathcal{N}(0, \sigma_2)) \leq C \Big| \frac{\sigma_1}{\sigma_2} - 1 \Big|, \tag{4.4}$$

for some constant $C > 0$, where by d_{TV} we denote the total variation distance between two probability distributions. A proof of this statement can be found in [19]. An exact formula is given in [33], however we find this expression more illustrative. The total variation distance between two probability distributions (or more precisely $1 - d_{TV}$) gives the overlap between them. Since we are considering mixtures of normals it seems sensible to keep the ratio between the standard deviations at two adjacent temperatures constant in order to have roughly the same overlap between adjacent temperatures, which is achieved by using a geometric progression.

For the standard deviation σ and the temperatures $\beta_1, ..., \beta_N$ we will consider three different scenarios, which we will describe below.

Between two swap moves we use a standard Metropolis-Hastings sampler for each individual replica using a normal random walk proposal with variance $0.1 \cdot \frac{\sigma}{\sqrt{\beta_i}}$ at the i^{th} temperature. We perform 5 Metropolis-Hastings steps between two swap attempts and

10000 swap attempts for a total of 50000 steps. We run each algorithm 100 times in order to obtain results which are less prone to randomness.

As the starting state for each replica in each algorithm we will choose the same mode $(-3, 0, 3, 6)$. We do this instead of, say, starting every replica in $(0, 0, 0, 0)$, which might seem to be a more natural choice, since this way every replica would move towards a random mode, since they are all symmetric around $(0, 0, 0, 0)$. The performance would depend on the intial modes each replica randomly moved to, and we therefore initialize every replica in the same mode.

We consider standard parallel tempering (hereafter PT, Algorithm 1), lifted parallel tempering (hereafter LPT, Algorithm 3) and lifted parallel tempering with circle (hereafter LPTC, Algorithm 4). For each of these algorithms we consider proposing one or two swaps during each swap attempt (Algorithm 5 and 6 for LPT and LPTC respectively; we will refer to the algorithms proposing two swaps as PT2, LPT2 and LPTC2).

4.4. Scenario 1: 20 temperatures

We first consider the case when the standard deviation is $\sigma = 0.5$ and we have 20 temperatures given by $\beta_i = 0.8^{20-i}$, $i = 1, ..., 20$. With this choice we ensure that on the one hand the chains at the highest temperature can comfortably traverse the modes (at the highest temperature the standard deviation of each mixture component is $\frac{0.5}{\sqrt{0.8^{19}}} = 4.17$), and on the other hand there are not too many temperatures where this is possible, as then the highest temperatures would be unnecessary and having fewer temperatures would increase performance.

We see in figure 4.1 that PT discovers modes faster than its non-reversible counterparts for each number of swaps per attempt. Further, performing two swaps indeed increases the numbers of modes discovered for each algorithm. However, we observe that between 1 and 3000 steps the non-reversible algorithms have discovered more modes than PT. This is in accordance with the intuition we provided in section 2.1 that in PT the diffusive random walk behavior of the replicas in the temperature space is especially an issue when the number of temperatures is large compared to the number of steps. At a low number of steps the persistency introduced by the lifting variables leads to a faster discovery of modes, whereas if the number of steps becomes large the replicas will also be able to oscillate between the two extreme temperatures in PT; since all replicas are performing this lazy random walk, this eventually leads to faster discovery of modes compared to the non-reversible algorithms.

We also note that LPT and LPTC are very similar in terms of the number of modes discovered. However, when only considering the first 3000 iterations it seems that LPTC is performing slightly worse than LPT. We will go into detail on this issue in the next section.

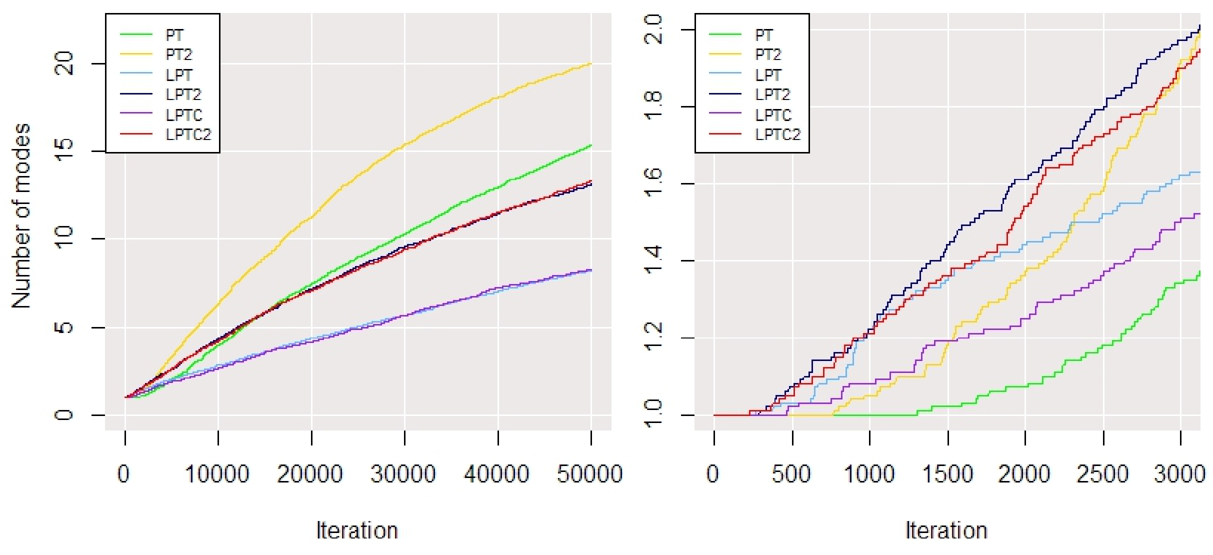

Figure 4.1.: Number of modes discovered by the six algorithms for $\sigma = 0.5$ and 20 temperatures $0.8^{19}, \ldots, 0.8^{0}$; average over 100 runs. Left for 50000 steps, right zoom-in for 3000 steps

Finally we also observe that the pace at which new modes are discovered seems to slow down slightly for all algorithms. This makes perfect sense, since the more modes have already been discovered the more likely it becomes that a new mode, which is transported up to temperature N, is in fact a mode that has already been seen before.

Next we inspect the number of round trips performed by the algorithms.

Algorithm	PT	PT2	LPT	LPT2	LPTC	LPTC2
Number of round trips	4.34	13.84	5.97	11.82	4.21	9.13

Table 4.1.: Number of round trips for the six algorithms; average over 100 runs. For the round trip rate divide by 50000

PT2 has a higher round trip rate than its non-reversible counterparts. This makes sense in light of the fact that the standard algorithms do discover more modes. While LPT and LPT2 only move one and two replicas respectively, PT/PT2 move all replicas. While each replica takes much longer to complete a round trip, all/many replicas eventually complete a round trip leading to a high total number of round trips. This also explains why, while PT2 performs slightly more round trips than its non-reversible counterparts, PT has slightly less round trips with an average of 4.34 than LPT and LTPC. Since PT only proposes one swap instead of two in PT2, a larger number of steps is necessary in order for the number of round trips to catch up with the non-reversible algorithms.

LPT2 and LTPC2 perform roughly twice as many round trips as their counterparts with only one copy of the lifting variable, which is the best we can hope for, since we expect some "friction" when the two copies of the lifting variable meet. In the next scenario we will observe that this is not always the case.

The round trip rates of LPTC and LPTC2 are lower than their counterparts LPT and LPT2, although the number of modes discovered are very similar. This is due to the fact that in LPTC and LTPC2 the current state can jump from $N-1$ to 1. When this happens a new state is swapped up to temperature N, but now the algorithms move a different replica; thus a (potentially) new mode was transported to temperature N, but only half a round trip was completed, thus leading to a lower number of round trips when discovering the same number of modes.

We provide an intuition for the observation that at a small number of steps the non-reversible algorithms discover modes faster whereas the standard algorithms perform better when the number of steps is large. We therefore investigate the current states in LPT, which coincides with the path of replica 1 in the temperature space.

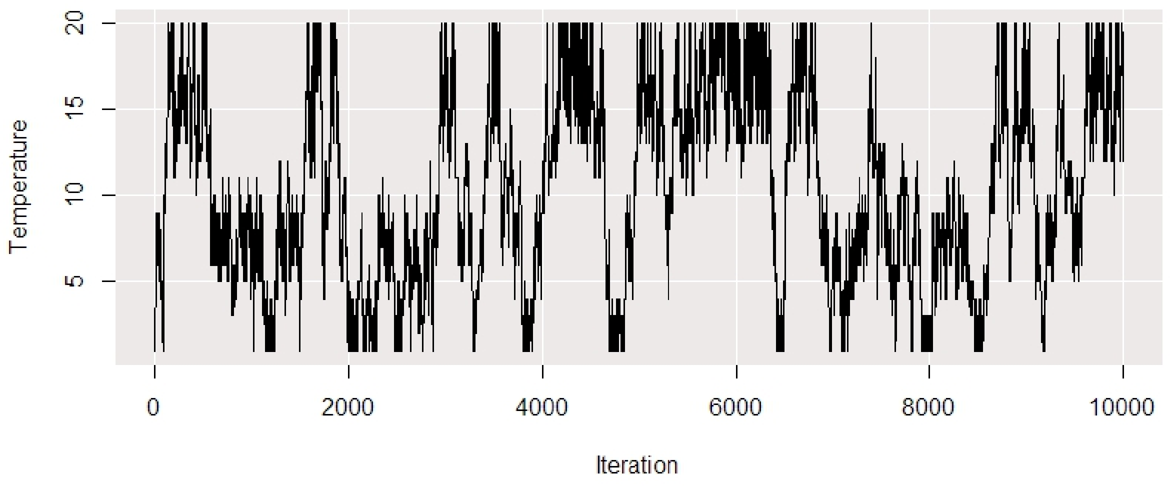

Figure 4.2.: Current state in LPT; this is also the trace of replica 1 in the temperature space

We see that in LPT replica 1 moves up and down the temperature space quite frequently. However, we observe that replica 1 often stays between temperatures 10 and 20, between 5 and 10 or between 1 and 5. This indicates that there are bottlenecks around temperatures 5 and 10, between which the current state spends long periods of time. We note that these correspond to standard deviations of around 1, which is roughly when the target distribution starts becoming flat enough to enable movement between the modes. While LPT spends its swap attempt to overcome these bottlenecks, PT keeps swapping other random replicas; thus each replica keeps performing a lazy random walk regardless of these bottlenecks. If the number of steps is small compared to the number of temperatures, LPT will overcome these bottlenecks and discover new modes, while PT does not. If however the number of steps is sufficiently large, PT will also eventually overcome these bottlenecks and multiple replicas will be able to transport new modes to the lowest temperature.

Next we inspect the traceplots of the four components of the chain at the highest temperature. Since they look very similar for one and two swaps per attempt we only provide the traceplots for the algorithms performing one swap in appendix B.1. For PT we see that new modes are frequently discovered. While new modes are also discovered for LPT, we do observe the baseline mode behaviour anticipated in section 3.2, the chain at temperature N always returns to $(-3, 0, 3, 6)$, the mode it started in. Allowing the current state to move between temperatures 1 and $N-1$ in LPTC indeed resolves this issue.

The autocorrelation plots are highly misleading, since the algorithms do not move freely between the modes. The autocorrelation function falls off very fast for LPT because of the baseline mode behaviour. Since the vast majority of iterations are spent in the same mode, it is only the local autocorrelation function in this one mode which falls off quickly, i.e. within this one mode the autocorrelation time is short, whereas this is not the case across all different modes. Therefore the autocorrelation functions fall off more slowly for PT and LPTC, which both spend more time in different modes and "acknowledge" the existence of multiple modes. This can be seen from the averages of the estimates for $\mu_{1:4}$ in the three algorithms. In LPT, the averages are very close to -3, 0, 3 and 6 respectively, whereas in PT and LPTC this is not the case, showing that, indeed, more time is spent in modes other than $(-3, 0, 3, 6)$ compared to LPT.

4.5. Scenario 2: 40 temperatures

In the second scenario we increase the number of temperature to 40, since the result of the previous scenario suggest that the non-reversible algorithms will compare more favorably against the standard algorithms if the number of temperatures is large compared to the total number of steps. We take 40 temperatures $\beta_i = 0.8^{40-i}$, $i = 1, ..., 40$ and decrease the standard deviation to $\sigma = 0.05$ so that the added temperatures are not higher than necessary. This essentially means that the modes are further apart from each other than for $\sigma = 0.5$, that is the target distribution is "more difficult".

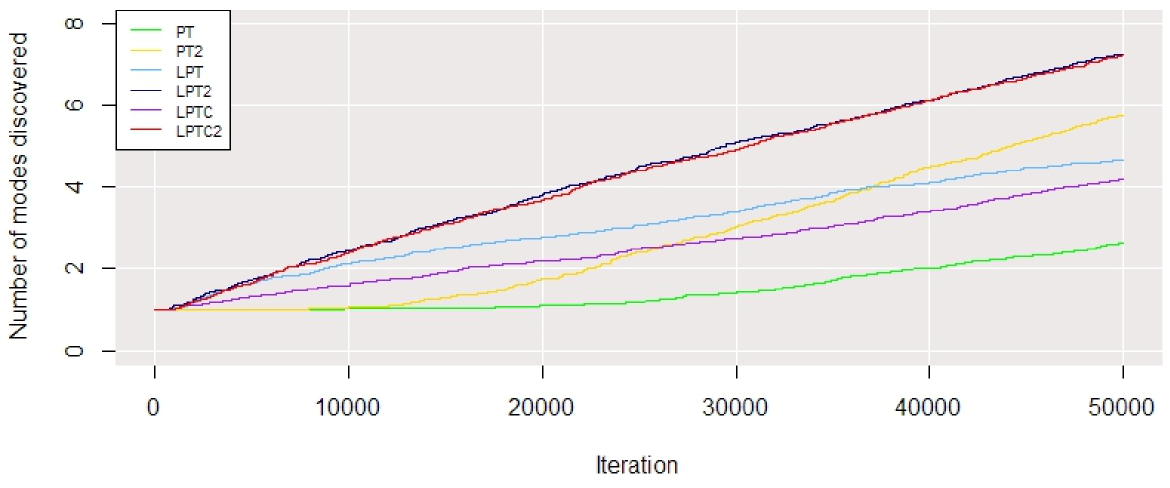

Figure 4.3.: Number of modes discovered by the six algorithms in 50000 steps for $\sigma = 0.05$ and 40 temperatures $0.8^{39}, ..., 0.8^{0}$; average over 100 runs

Indeed, in this case the non-reversible algorithms discover modes faster than PT for both one and two swaps per attempt, thus 40 temperatures seem to be what we informally call "many" temperatures for $K = 10000$ swap attempts. We also note that the difference between the PT and LPT/LPTC is largest around $20000 - 25000$ steps and between PT2 and LTP2/LPTC2 at around $10000 - 15000$ steps. Firstly, this indicates that, indeed, the reversible algorithms start off discovering modes more slowly than its non-reversible counterparts, but after a sufficient number of steps increase their speed at which modes are discovered. Secondly, this "speed-up" begins earlier for PT2 than for PT, which is to be expected since two swaps instead of one are performed, that is the replicas move more frequently in PT2.

While in the previous scenario LTP and LPTC discovered modes at a very similar pace for both one and two swaps per attempt, this is only the case for LTP2 and LTPC2 here; LPTC discovers new modes slightly more slowly than LPT. In order to try to explain this phenomenon we inspect the trace plot of the current state in LPTC, addressing an issue mentioned in the previous section.

As in the previous section, figure 4.4 suggests the existence of two bottlenecks around temperatures 5 and 10. This again corresponds to temperatures where it traversing the modes starts to become likely. Between steps 1500 and 5000 the current state never entirely crossed this region but always turned around at some point. This is also the reason why we suspect that LTPC performs worse than LTP, since as long as the bottleneck is not overcome, no new modes from temperature 1 can be transported to temperature N. When the current state is at 1 in LPT, it will approach this bottleneck as long as it is stuck at temperatures $1 - 4$. In LTPC however the current state can jump to temperature $N - 1$

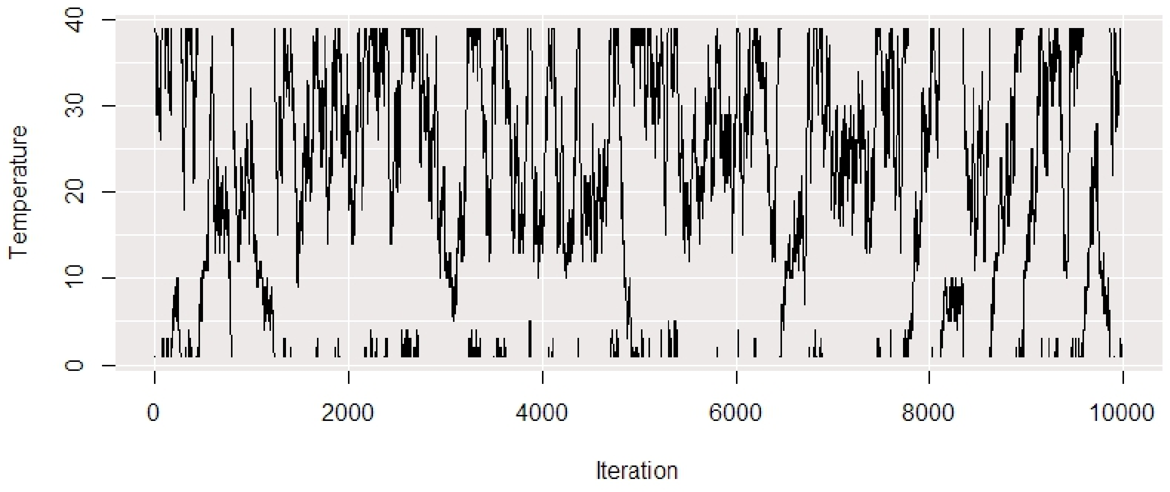

Figure 4.4.: Current state in LPTC; this is not the trace of replica 1 in the temperature space anymore, since the "active" replica changes when moving from $N-1$ to 1 and vice versa

from 1, and move at those low temperatures. It then approaches the bottleneck from the other side, and if it is rejected it can again move back to $N-1$ and jump to 1. The issue is that this way the bottleneck is tackled less often and thus it takes a comparably long time to overcome it. This was less a problem in the previous scenario as there were only 20 temperatures. Even if the current state jumped to temperature 19 it would approach the bottleneck from above rather soon. This is not the case here since there are 20 additional temperatures, i.e. the current state can move around temperatures $20-39$ without approaching the bottleneck. This is another aspect of the lost persistency of LPTC compared to LPT, it does not try to overcome bottlenecks as persistently.

This seems to be less an issue for LPTC2 since there are two copies of the lifting variable, which act like a boundary for each other. Thus, if both of them are at temperatures $20-40$, one of the copies will "push" the other copy down towards the bottleneck.

Next we present the number of round trips each algorithm performed in this setting.

Algorithm	PT	PT2	LPT	LPT2	LPTC	LPTC2
Number of round trips	0	0.1	2.63	4.44	1.29	1.74

Table 4.2.: Number of round trips for the six algorithms; average over 100 runs. For the round trip rate divide by 50000

In this scenario, overall less modes were discovered and round trips performed compared to the last scenario, since we decreased the standard deviation by the factor 10 and kept the number of steps fixed. The round trips performed by PT and PT2 were decreased drastically, PT did not perform any round trips at all and PT2 only one round trip every ten runs. However, both algorithms do still discover modes, since there are replicas which perform half a round trip, bringing a new mode to temperature N.

We further observe that in this case LPT2 and LPTC2 do not perform twice the number of round trips of LPT and LPTC. If both have the same current direction when they meet, it might happen that they "stick together" (see figure 4.5). This is a very minor issue, since we only observed this to happen for a couple of steps only and never over a long period of time. However, what we occasionally observed happening is that replica 1 is handed over from the first lifting variable to the second and then back from the second to the first; if the two lifting variables separate like this, replica 1 will travel back to temperature 1 instead of N, thus delaying round trips. This is more severe in this case compared to the previous scenario as, since there are more temperatures, it will take longer until the two replicas meet again. However, the performance of LPT2 and LTPC2 is still better both in terms of number of modes discovered and round trip rate than introducing two copies of the lifting variable operating independent of each other.

Given the overall lower number of modes discovered we provide the traceplots of the mean estimates μ and the autocorrelation for the algorithms performing two swaps in appendix B.2. The figures are very similar to the figures in the previous section. The only major difference is that PT2 also shows a baseline mode behaviour, which can be explained by the added temperatures. Around temperatures $20 - 40$ new modes are only discovered if they are swapped there from higher temperatures, therefore we observe that at these temperatures eventually only four replicas are in different modes than the initial one, leading to the majority of time being spent in the initial mode. This also leads to the autocorrelation function of PT falling off relatively quickly.

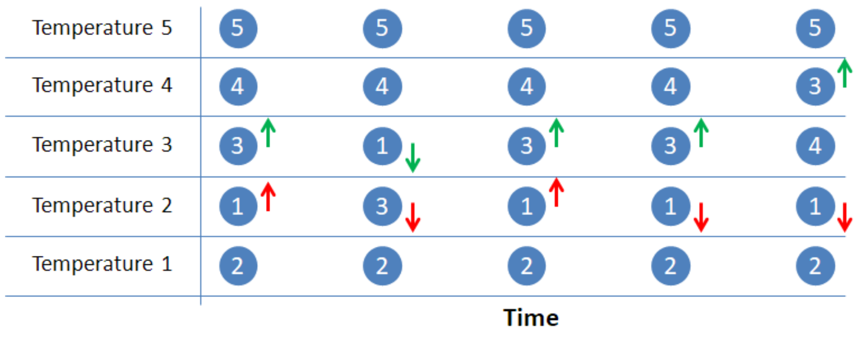

Figure 4.5.: Example with 5 temperatures. The two copies of the lifting variable stick together, and after seperating replicas 1 and 3 move back to where they came from.

4.6. Scenario 3: Adjusting the acceptance probability

So far we have chosen a set of temperature which ensures that the chains at the highest temperatures can comfortably traverse the modes and there are not to many unnecessarily high temperatures. The choice of $q = 0.8$ was, while good for illustrative purposes, rather arbitrary. In this section we try to choose a temperature set which is optimal for PT and compare the algorithms in this setting. In order to do so we consider the acceptance probability of attempted swaps in PT.

In [33], Rathore et al. studied the acceptance rate of swap attempts on a Go-type Protein model and the Lennard-Jones fluid and the optimal acceptance rate was found to be 20%. The optimal acceptance rate of 23% obtained in [21] is remarkably similar to that value.

We expect the acceptance probability of swaps to depend primarily on q if we choose N temperatures $\beta_i = q^{N-i}$, $i = 1, ..., N$, largely independent of N and the standard deviation σ, since equation 4.4 indicates that the overlap between two adjacent distributions and therefore the acceptance probability between the replicas at adjacent temperatures can be bounded by a term only depending on the ratio of the variances. [This paragraph unnecessary?]

In order to achieve acceptance probabilities of the magnitude suggested above, we experimentally found that we would roughly have to choose $q = 0.25$. However, if we fix the standard deviation of $\sigma = 0.05$, only 7 temperatures would be necessary in order to enable the chain at the highest temperature to traverse the modes comfortably. The results in section 4.4 already suggest that in such a setting PT/PT2 will perform much better than the non-reversible algorithms if we performed $K = 10000$ swap attempts. If we kept using 40 temperatures, we would need to decrease the standard deviation to $\sigma = 7 \cdot 10^{-12}$ in order to ensure that the highest temperatures are not unnecessary. With such a small standard deviation we found that mostly only one or no new mode was found in a reasonable number of steps.

We therefore keep the standard deviation $\sigma = 0.05$. Choosing $q = 7$ would lead to less than 10 temperatures, that is only a quarter of the number of temperatures considered in the previous scenario, where we found $K = 10000$ to be small compared to $N = 40$ temperatures. By the intuition given in section 2.1 this would imply that we should decrease the number of steps to less than $\frac{10000}{4^2} = 625$. At such a small number of steps all algorithms, again, either found one or no new mode; comparing such numbers does not seem too meaningful.

Instead of considering the number of modes discovered we consider the time each algorithm needs to discover the first new mode. This enables us to compare the algorithms in a regime where we expect them to find only one or no new modes, that is we can try to find temperatures which are optimal for PT and compare the algorithms at a, relative to the number of temperatures, small number of steps, where we expect the advantages of the non-reversible algorithms to show.

Rathore et al. note in [33] that the acceptance probability should be made equal between all adjacent pairs of temperatures in order to ensure that each replica spends roughly the same time at each temperature. To do so they start with one temperature and iteratively determine the next temperature which yields the same acceptance probability for a swap.

Since the optimal choice of temperatures is not the main purpose in this dissertation we stick with a simple geometric progression, adapting the acceptance probability by choosing different values for q. We do however address the issue of the existence of bottlenecks, which seem to appear at temperatures giving a standard deviation around 1, which was discussed in the previous sections.

We therefore choose two different values q_1 and q_2 and consider temperatures of the form

$$\beta_i = \begin{cases} q_1^{N_1-i} q_2^{N_2} & \text{for} \quad i = 1, ..., N_1 - 1 \\ q_2^{N_2-(i-N_1)} & \text{for} \quad i = N_1, ..., N_1 + N_2, \end{cases} \tag{4.5}$$

that is we change the rate for the geometric progression from q_2 to a higher value q_1 in order to account for the observed lower acceptance probability at the highest temperatures. We then run PT at temperatures $1, ..., N_1$ and $N_1 + 1, ..., N_1 + N_2$ separately, each 100 times for $K = 1000$ swap attempts and choose values for q_1 and q_2 so that the observed acceptance probability of swaps is the same in both regimes. We do this for 4 different temperature sets and measure the time PT needs to discover the first new mode.

Parameters		Acceptance probability	Time to first mode (in seconds)
q_1	0.45	0.14	23.65
q_2	0.2		
N_1	6		
N_2	3		
q_1	0.6	0.21	21.62
q_2	0.25		
N_1	7		
N_2	4		
q_1	0.67	0.36	26.45
q_2	0.35		
N_1	9		
N_2	5		
q_1	0.77	0.53	43.98
q_2	0.5		
N_1	12		
N_2	8		

Table 4.3.: Acceptance probability and median time to the discovery of the first new mode obtained from 100 runs of PT with $K = 1000$ swap attempts each

We see that, indeed, the temperatures we used in the previous sections do not seem to be optimal for PT, and that the suggested acceptance probability of $20 - 23\%$ seems to give the best results; in our case choosing temperatures as in 4.5 with $q_1 = 0.6$, $q_2 = 0.25$, $N_1 = 7$ and $N_2 = 4$ gives an acceptance probability of 21% and with 21.62 seconds the lowest time to the discovery of the first new mode.

Using these temperatures we compare our non-reversible algorithms with PT and PT2 and report the median time to the discovery of the first new mode obtained in 100 runs.

Algorithm	PT	PT2	LPT	LPT2	LPTC	LPTC2
Time to first mode in seconds	21.53	14.53	17.45	12.50	30.75	19.15

Table 4.4.: Median time to the discovery of the first new mode over 100 runs for the six algorithms in seconds

These results confirm that LPT and LTP2 perform (slightly) better than PT and PT2 respectively when the number temperatures is large compared to the number of steps. We used the same temperatures for all six algorithms, which were chosen based on the performance of PT, so there might be better choices for LPT and LPT2. The other temperatures considered in table 3.3 did not give better results for LPT, although they were still better compared to PT at the same temperatures. However, there might be entirely different choices for the temperatures which give better performances for the non-reversible algorithms.

LTPC and LTPC2 have a worse performance than their counterparts without the circle, which is reasonable, since adding the possibility of moving the current state from temperature 1 to $N - 1$ and vice versa allows the chain at temperature N to stay in new modes at the cost of persistency introduced by the lifting variable, since a replica is not pushed to the highest and lowest temperatures as determinedly as in LPT.

In summary these three examples show that the non-reversible algorithms are performing better than PT/PT2 in terms of number of modes discovered and number of round trips when the number of temperatures is large relative to the number of steps taken. When considering the time to the discovery of the first new mode at temperature N we are technically enforcing a small number of steps, since we are only running the algorithms until the first new mode is discovered. Even when choosing the temperatures best for PT, LPT and LPT2 discover the first mode faster than their counterparts PT and PT2 respectively.

5. Application to a Bayesian Nonparametric Latent Feature Model

In this chapter we present the bayesian nonparametric latent feature model following [28], introduce the Indian Buffet Process as in [13] and [14], explain how one can use the Gibbs sampler to conduct inference in such a model based on [10] and [28] and lastly compare parallel tempering to the Gibbs sampler in this model.

Assume we observe some data $\mathbf{X}$ and would like to conduct inference some unknown parameter ϕ, which are related to the observations via a likelihood function $p(\mathbf{X}|\phi)$. In a Bayesian parametric framework we assume ϕ to be a finite dimensional real-valued vector, put our prior beliefs about the parameter into a prior $p(\phi)$ and update our beliefs after observing the data $\mathbf{X}$ to the posterior distribution $p(\phi|\mathbf{X}) \propto p(\phi)p(\mathbf{X}|\phi)$. However, with enough data any parametric model becomes too simplistic and will be rejected ([8]). One way to address this issue is to fit multiple models of different complexity and carry out model selection, where the goal is to obtain a well-fitting model while having a reasonable complexity. In the nonparametric setting we put priors on an infinite dimensional parameter ϕ. By doing so our model can grow in complexity and adapt to any number of observations.

Defining reasonable priors in the nonparametric setting is not trvial. Gaussian processes are a widely used model and put a prior distribution on the space of functions. This can be directly applied to regression and classification problems among others, see e.g. [32]. The Dirichlet Process is another popular class of stochastic processes which puts a distribution on the space of discrete probability measures. It is often used for density estimation and clustering via mixture models. Thus it is an often used prior in latent class models, where we assume that each observation x_i belongs to an unobserved class z_i, which determines the generative model of the data. The Chinese Restaurant Process is an intuitive way to construct the Dirichlet Process, for details see e.g. [40].

The Indian Buffet Process puts a prior on binary matrices and plays the same role for the Beta Process, which was introduced in [16] and is a distribution on discrete measures, as the Chinese Restaurant Process does for the Dirichlet Process. This is the prior we will use in the following in the latent feature model, where we will work with the Indian Buffet Process representation.

5.1. Latent Feature Model

We present the bayesian nonparametric latent eature model following [28].

Assume we have n observations $\mathbf{X} = (x_1, ..., x_n)^T \in \mathbb{R}^{n\times p}$ and ϕ, some parameter related to the data via a likelihood function $p(\mathbf{X}|\phi) = \prod_{i=1}^n p(x_i|\phi)$. In the Gaussian mixture model considered in chapter 4, which is a specific case of latent class models, we assumed that each observation x_i was generated from one of four normal distributions, that is in addition to the means and standard deviations each observation depends on the class z_i indicating from which mixture component the observation was drawn. In this case we can think of z_i either as an integer between 1 and 4 or as an 1×4 binary vector with exactly one entry being equal 1, indicating to which class this observation belongs.

In the latent feature model we assume that there is a $1 \times K$ binary feature vector z_i for each observation x_i indicating which features individual i possesses, where K is the total number of features. Unlike in the latent class model z_i can now have multiple or no entries equal 1. In matrix notation the observations $\mathbf{X}$ depend on the binary feature matrix $\mathbf{Z} \in \{0,1\}^{n\times K}$ via the likelihood $p(\mathbf{X}|\phi, \mathbf{Z})$.

In fact, we could represent a latent feature model with a latent class model and vice versa by identifying each combination of features with one class; that is, for K features we would have 2^K classes. It is clear that the latent feature model provides a more compact representation of the structure if it is appropriate. Also, in latent class models we might have very little or even no observations for a class, making it hard to make predictions for a new individual of that class. If we wanted to predict x for an individual with features z in a latent feature model we do not need to have observations with this exact combination of features, but the effect of each feature can be learned from our available observations. However, if the latent class model explains the data well, it might be more efficient than a latent feature model.

Since the number of classes K is unknown we would like to have a prior on all binary matrices $\mathbf{Z} \in \{0,1\}^{n\times K}$ for all $K \in \mathbb{N}$. The prior should not depend on the ordering of the features or of the observations, so $p(\mathbf{Z}) = p(\mathbf{Z'})$ should hold when $\mathbf{Z'}$ can be obtained from $\mathbf{Z}$ by permuting the columns and/or rows. One could also consider non-negative integer valued feature matrices, for which the Gamma Proces provides a prior which is presented in [42]. We will only consider binary matrices and the Indian Buffet Process, which provides us with an appropriate prior.

5.2. Indian Buffet Process

We present the Indian Buffet Process following [13] and [14].

Imagine n customers enter a restaurant with infinitely many dishes arranged along a line, one after another. The first customer tries the first Poisson(α) dishes, where $\alpha > 0$ is a parameter if this process. When the i^{th} customer enters the restaurant, he first tries every dish which has previously been chosen by another customer with some probability proportional to the dishes popularity; customer i tries dish k with probability $\frac{m_k}{i}$, where m_k is the number of customers before i who have tried dish k. He then moves on and tries Poisson($\frac{\alpha}{i}$) new dishes.

Doing this we obtain a binary matrix $\mathbf{Z}$ with n rows and infinitely many columns, where $z_{ik} = 1$ indicates that customer i has tried dish k. With probability 1 $\mathbf{Z}$ will only have finitely many non-zero entries, hence we can ignore the zero columns and consider the $n \times K$ matrix $\mathbf{Z}$, where K is the number of dishes which have been tried by at least one customer.

However, if $\mathbf{Z}$ is generated as described above, neither the customers (rows) nor the dishes (columns) are exchangeable. Therefore consider the left-ordered form of a binary matrix, where we sort the columns in descending order from left to right according to the binary number represented by that column, where the first row corresponds to 2^0, the second row to 2^1 and so on. Many binary matrices have the same left-ordered form, therefore this defines an equivalence class on the space of binary; two binary matrices $\mathbf{Z} \sim \mathbf{Z}$' are in the same equivalence class, denoted $[\mathbf{Z}]$, if they have the same left-ordered form. The Indian Buffet Process defines a an exchangeable prior on these equivalence classes.

The probability of an equivalence class $[\mathbf{Z}]$ under the Indian Buffet Process with parameter α is given by

$$p([\mathbf{Z}]) = \frac{\alpha^K}{\prod_{h=1}^{2^N-1} K_h!} \exp(-\alpha H_n) \prod_{k=1}^{K} \frac{(n-m_k)!(m_k-1)!}{N!}, \tag{5.1}$$

where K is the number of non-zero columns of $\mathbf{Z}$, K_h the number of columns which are the binary representation of the number h, m_k the number of customers who tried dish k, i.e. the number of ones in the k^{th} column, and $H_n = \sum_{i=1}^{n} \frac{1}{n}$ the n^{th} harmonic number. Clearly these values are independent under permutations of the columns, that is they are the same for all elements in $[\mathbf{Z}]$. A derivation of this expression can be found in [14].

In the following we will write IBP(α) for the Indian Buffet Process with parameter α.

The Indian Buffet Process introduced above depends only on one parameter α, which controls both how many features are possessed by at least one individual, i.e. how many non-zero columns the feature matrix $\mathbf{Z}$ has, and how many features each individual has, i.e. the number of ones in each row in $\mathbf{Z}$. A two-parameter extension of the Indian Buffet Process, where these two quantities can be controlled separately, is presented in [10].

Finally we briefly address the connection between the Indian Buffet Process and the Beta Process. Since the rows of the feature matrix $\mathbf{Z}$ are exchangeable, de Finetti's theorem implies that

$$p(\mathbf{Z}) = p(z_1, ..., z_n) = \int \Big(\prod_{i=1}^{n} p(z_i|B)\Big) dP(B) \tag{5.2}$$

holds, where B is a random variable conditional on which the z_i are independent, and P is its distribution. Thibaux and Jordan showed in [41] that for the Indian Buffet Process this distribution, which they refer to as "de Finetti mixing distribution", is given by the Beta Process, which was introduced in [16]. This link can be used to derive similar extensions for the Beta Process as for the Dirichlet Process, which is the de Finetti mixing distribution for the Chinese Restaurant Process. For instance, analogously to the Hierarchical Dirichlet Process in [39], Thibaux and Jordan derived the Hierarchical Beta Process in [41], which can be used for example as prior in document classification problems. We will not go into further detail about the Beta Process, as we will work with the IBP representation. A good summary can be found in [28].

5.3. Inference algorithms for latent feature models

In this section we will present how inference can be conducted in the latent feature model with the IBP as prior for a generic likelihood. In particular we will consider the Gibbs sampler, which has also been used in [10], [13], [14] and [28].

Wood and Griffiths suggested the use of a particle filter in [46]. Teh et al. proposed a slice sampler in [38], which uses the stick-breaking construction for the IBP, another parallel to the Dirichlet Process. An improved version of Gibbs sampling is shown by Doshi-Velez and Ghaharmani in [44]. We will only consider the classic Gibbs sampler.

An issue in these kinds of models is the fact that the target distribution is highly multimodal and the Gibbs sampler can get stuck in a local optimum easily. We therefore propose using parallel tempering and compare its performance with the Gibbs sampler on a toy example using simulated data.

5.3.1. Gibbs sampler in the latent feature model

Given some observed data $\mathbf{X} = (x_1, ..., x_n)^T$, the goal is to draw samples from the posterior distribution $p(\mathbf{Z}, \phi|\mathbf{X}) \propto p(\mathbf{X}|\mathbf{Z}, \phi)p(\mathbf{Z})p(\phi)$, where $\mathbf{Z}$ is the unobserved latent feature matrix, ϕ some parameter and $p(\mathbf{Z})$ and $p(\phi)$ priors we assume to be independent; in particular we will use the IBP with parameter α as prior for $\mathbf{Z}$. We will closely follow [28] in presenting the Gibbs sampler.

Given a current parameter vector $\phi = (\phi_1, ..., \phi_d)$ and a $n \times K$ feature matrix $\mathbf{Z}$ we iteratively do the following updates, where de denote by $\mathbf{Z}_{-ik}$ and ϕ_{-j} all of $\mathbf{Z}$ but the $(i, k)^{th}$ entry and all of ϕ but the j^{th} entry respectively.

1. for $i = 1, ..., n$, $k = 1, ..., K$ sample

$$z_{ik} \sim p(z_{ik}|\mathbf{Z}_{-ik}, \phi, \mathbf{X}) \propto p(X|\mathbf{Z}_{-ik}, z_{ik}, \phi)p(z_{ik}|\mathbf{Z}_{-ik}).$$

2. for $i = 1, ..., n$ sample K_i^{new}, the number of new features associated with row i, from

$$p(K_i^{new}|\mathbf{X}, \mathbf{Z}, \phi, \alpha) \propto p(\mathbf{X}|\mathbf{Z}, K_i^{new}, \phi)p(K_i^{new}|\alpha)$$

   and add K_i^{new} columns containing all zeros but a one at the i^{th} entry to $\mathbf{Z}$.

3. for $j = 1, ..., d$ sample

$$\phi \sim p(\phi_j|\phi_{-j}, \mathbf{Z}, \mathbf{X}) \propto p(\mathbf{X}|\mathbf{Z}, \phi_{-j}, \phi_j)p(\phi_j|\phi_{-j}).$$

Step 3 is sampling the parameter ϕ given the data matrix $\mathbf{X}$ and a feature matrix $\mathbf{Z}$ and therefore application specific. We will address this step in section 5.4 when we introduce the model we will implement the algorithm on.

Steps 1 and 2 is updating the feature matrix $\mathbf{Z}$ given $\mathbf{X}$ and ϕ. For step 1 we need to evaluate the expression for z_{ik} equal 0 and 1 and normalize these terms. $p(X|\mathbf{Z}_{-ik}, z_{ik}, \phi)$ is just the likelihood given both the parameters and latent features. For the second term we can use the exchangeability of the IBP prior and assume that the i^{th} row corresponds to the customer, who enters the restaurant last. As described in section 5.2 the probabilites are

$$\begin{aligned} p(z_{ik} = 0|\mathbf{Z}_{-ik}) &= \frac{n - m_{-ik}}{n} \\ p(z_{ik} = 1|\mathbf{Z}_{-ik}) &= \frac{m_{-ik}}{n}, \end{aligned}$$

where by $m_{-ik} = \sum_{j \neq i} z_{jk}$ we denote the sum of the k^{th} column of $\mathbf{Z}$ excluding the i^{th} entry, that is how many of the other customers tried dish k.

For step 2 the term $p(\mathbf{X}|\mathbf{Z}, K_i^{new}, \phi)$ is in general not easy to evaluate, since we only store the finitely many parameters ϕ associated with features which are possessed by at least one individual i. In the above likelihood the parameters ϕ correspond to the features $\mathbf{Z}$, but we have not yet sampled any parameters corresponding to the newly added K_i^{new} features. These have to be integrated out, which is easily doable if we have a conjugate prior. For the IBP prior introduced in section 5.2 the prior probability $p(K_i^{new}|\alpha)$ is Poisson $\left(\frac{\alpha}{n}\right)$. In principle we would have to consider $K_i^{new} \in \{0\} \cup \mathbb{Z}^+$. However, we cannot evaluate $p(\mathbf{X}|\mathbf{Z}, K_i^{new}, \phi)p(K_i^{new}|\alpha)$ for infinitely many values of K_i^{new}. We therefore sample K_i^{new} from $\{0, 1, ..., c\}$ for some large c, since $\lim_{c\to\infty} \mathbb{P}[K_i^{new} > c] = 0$.

Görür et al. developed an MCMC algorithm for an elimination by aspects model, which uses the IBP as prior for the features and a non-conjugate prior for the parameters ϕ; details can be found in [11]. This algorithm is very similar to algorithm 8 of [29], which is used for conducting inference in Dirichlet Process models with a non-conjugate prior. While the accelerated Gibbs sampler from [44] can only be applied in models with a conjugate prior, both the particle filter from [46] and the slice sampler from [38] can also be applied in the case of a non-conjugate prior.

Finally we note that typically the hyperparameter α is not known. We can also sample from α by augmenting above Gibbs sampler with a standard Metropolis-Hastings step for α.

5.3.2. Parallel Tempering in the latent feature model

One shortcoming of the Gibbs sampler is the fact that it has difficulties escaping from local optima. This is especially severe in the latent feature model, as the posterior distribution will be highly multimodal. This becomes particularly obvious if we think about latent classes instead of features as described in section 5.1, that is each particular combination of features corresponds to one class. Every assignment of features which puts the correct observations into the same class will be a local optimum. For example, if we consider four features, there will be at most $2^4 = 16$ classes; with five features we have $2^5 = 32$ possible classes, thus it is much easier to group the observations correctly with five features, and once it has done so the Gibbs sampler will be trapped in this local optimum.

In order to resolve this issue we will apply LPTC (Algorithm 4), where we will use the Gibbs sampler described above for the steps at fixed temperatures. Given temperatures $\beta_1 < ... < \beta_N = 1$ we will consider tempered distributions of the form

$$\pi_n(\mathbf{Z}, \phi) \propto p(\mathbf{X}|\mathbf{Z}, \phi)^{\beta_n} p(\mathbf{Z}) p(\phi) \tag{5.3}$$

as was done in [30] for Bayesian inference. Using this form we are giving the data $\mathbf{X}$ less influence on our prior at high temperatures.

When proposing to swap two replicas we evaluate the right hand side of 5.3 using equation 5.1 for the prior of the feature matrix $p(\mathbf{Z})$.

5.4. Example: Linear Gaussian likelihood

We will compare the Gibbs sampler and parallel tempering in a linear Gaussian latent feature model as described in [13] and [28].

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be our observation matrix. Each observation x_i has some underlying latent features z_i, which make up the rows of the feature matrix $\mathbf{Z}$. To each feature corresponds a parameter vector A_k, making up the columns of the parameter matrix $\mathbf{A}$; $\mathbf{A}$ plays the role of the parameter ϕ from above. Our generative model is then

$$\begin{aligned} p(\mathbf{Z}|\alpha) &= \mathrm{IBP}(\alpha) \\ p(A_k|\sigma_A^2) &= \mathcal{N}(0, \sigma_A^2 \mathbf{I}) \\ p(x_i|\mathbf{Z}, \mathbf{A}, \sigma_x^2) &= \mathcal{N}(z_i A, \sigma_x^2 \mathbf{I}), \end{aligned}$$

where α, σ_A^2 and σ_x^2 are hyperparameters to be inferred.

In this conjugate model the likelihood can be shown to be

$$\begin{aligned} p(\mathbf{X}|\mathbf{Z}, \sigma_x^2, \sigma_A^2) = & \frac{1}{(2\pi)^{\frac{nd}{2}} \sigma_x^{(n-K)d} \sigma_A^{Kd} \left| \det \left(\mathbf{Z}^T \mathbf{Z} + \frac{\sigma_x^2}{\sigma_A^2} \mathbf{I} \right) \right|^{\frac{d}{2}}} \\ & \exp \Big(- \frac{1}{2\sigma_x^2} \operatorname{tr} \Big(\mathbf{X}^T \Big(\mathbf{I} - \mathbf{Z} \Big(\mathbf{Z}^T \mathbf{Z} + \frac{\sigma_x^2}{\sigma_A^2} \mathbf{I} \Big)^{-1} \mathbf{Z}^T \Big) \mathbf{X} \Big) \Big), \end{aligned} \tag{5.4}$$

that is $\mathbf{A}$ can be integrated out completely and does not need to be considered in our inference algorithms. A derivation of this expression can be found in [14].

We would need to evaluate 5.4 everytime we update z_{ik} in the Gibbs sampler. This involves many computationally expensive matrix inversions, which can be avoided as in [14] due to the fact that we only change a single entry at a time. Define $\mathbf{M} = \left(\mathbf{Z}^T \mathbf{Z} + \frac{\sigma_x^2}{\sigma_A^2} \mathbf{I} \right)^{-1}$ and $\mathbf{M}_{-i} = \left(\sum_{j \neq i} z_j^T z_j + \frac{\sigma_x^2}{\sigma_A^2} \mathbf{I} \right)^{-1}$, then using $\mathbf{Z}^T \mathbf{Z} = \sum_{i=1}^n z_i^T z_i$ we have the identities

$$\mathbf{M}_{-i} = (\mathbf{M}^{-1} - z_i^T z_i)^{-1} = \mathbf{M} - \frac{\mathbf{M} z_i^T z_i \mathbf{M}}{z_i \mathbf{M} z_i^T - 1} \tag{5.5}$$

and

$$\mathbf{M} = (\mathbf{M}_{-i}^{-1} + z_i^T z_i)^{-1} = \mathbf{M}_{-i} - \frac{\mathbf{M}_{-i} z_i^T z_i \mathbf{M}_{-i}}{z_i \mathbf{M}_{-i} z_i^T + 1}. \tag{5.6}$$

This means that when only changing one row z_i of $\mathbf{Z}$ we can use these updates and instead of inverting the matrix in 5.4.

5.5. Comparison of the Gibbs sampler and parallel tempering

We apply both the Gibbs sampler and parallel tempering to a toy example using the linear Gaussian likelihood and $n = 50$ observations with $d = 10$ dimensions. We simulated the data using four latent features, that is we generated a 50×4 feature matrix $\mathbf{Z}$, where we set each entry to 1 with probability 0.5. As described in the previous section we generate parameters **A** as above with $\sigma_A^2 = 0.49$ and observations x_i with $\sigma_x^2 = 0.05$. For the parallel tempering algorithm we chose 20 temperatures $\beta_i = 0.98^{20-i}$, $i = 1, ..., 20$.

We initialize both algorithms with $K = 1$, $\alpha = 1$, $\sigma_A^2 = 1$ and $\sigma_x^2 = 1$ and sample each entry z_{i1} from $\text{Ber}(\frac{1}{2})$. We use the same initial feature assignments for all replicas in parallel tempering. We run the Gibbs sampler for 6000 and parallel tempering for 2000 iterations.

We observe that the Gibbs sampler is more likely to get stuck in a local optimum when the number of observations n is large, the dimension of each observation d is large or the variance in the generation of the data σ_x^2 is small. All three factors contribute towards the likelihood distinguishing more clearly between feature matrices **Z** which do or do not correctly classify the observations, thus making it harder to leave a feature representation with more features, which separates the observations correctly (i.e. a representation assigning the same features to observations with the same underlying features).

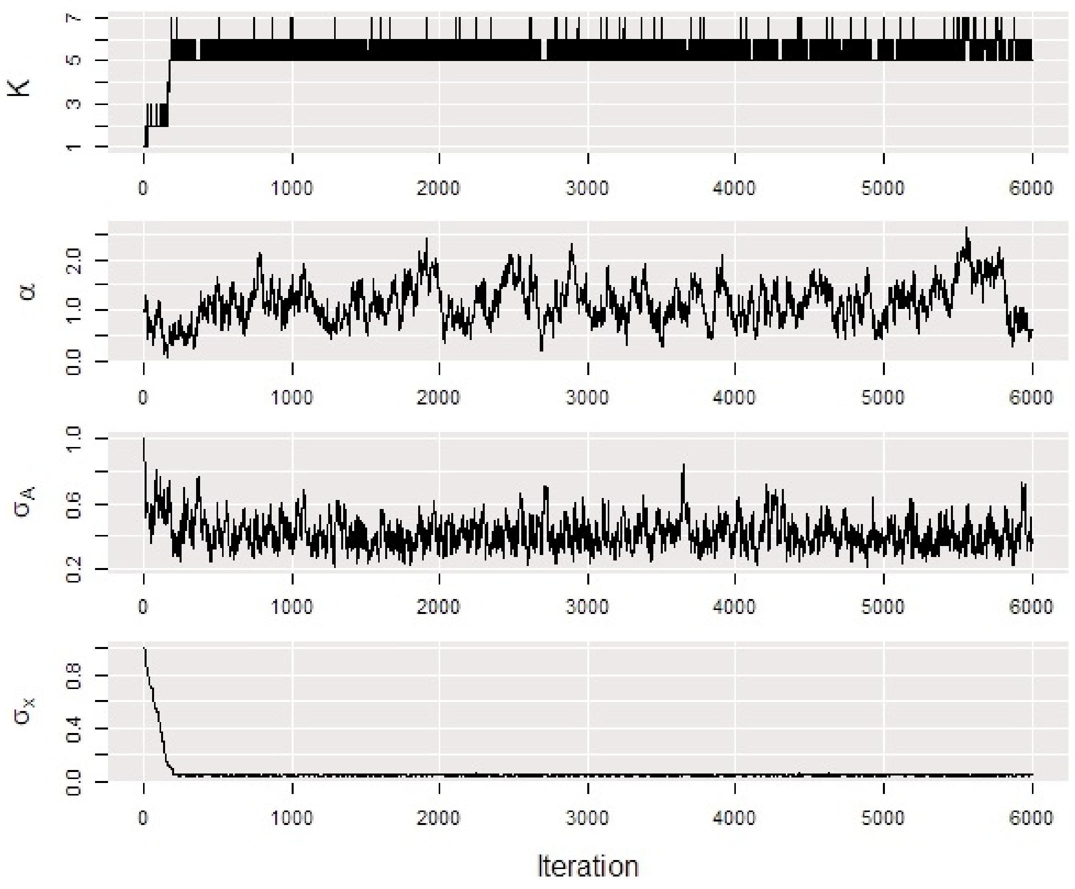

Figure 5.1.: Trace plots for the number of non-zero features K and the parameters α, σ_A and σ_x over 6000 iterations of the Gibbs sampler.

While in the Gibbs sampler the variance estimates converge very quickly to the true values of σ_x^2 and σ_A^2, the number of features K is constantly overestimated. Unlike in the example considered in [12], where the variance for the observations was higher with $\sigma_x^2 = 0.25$, it is not the case here that the correct number of features are possessed by many individuals and the other features only by one or two individuals. Inspecting the sequence of feature matrices shows that the Gibbs sampler stabilizes at five features occurring 10 to 30 times each, occasionally adding more features only occurring once or twice. This shows that, indeed, getting stuck in local optima is an issue when running inference algorithms for the latent feature model.

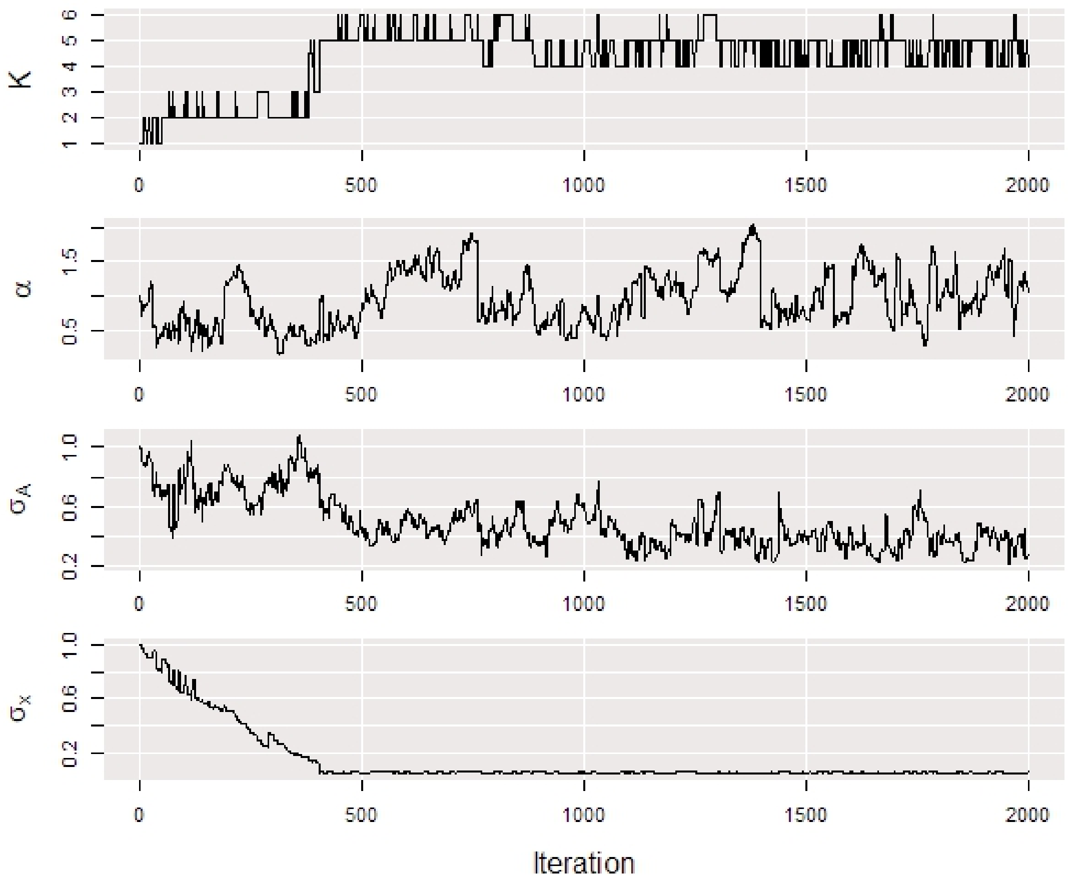

Figure 5.2.: Trace plots for the number of non-zero features K and the parameters α, σ_A and σ_x over 2000 iterations of parallel tempering (1000 swap attempts, 2 steps with the Gibbs sampler in between).

As in the Gibbs sampler the estimates for σ_x^2 and σ_A^2 converge very quickly to the true values. Initially, parallel tempering also finds a local optimum with five features, but the the 384^{th} swap (768^{th} iteration) successfully swaps the chains at temperatures 19 and 20 and brings the true configuration for $\mathbf{Z}$ to temperature 20.

Next we inspect the log-likelihood of the two algorithms against time, since parallel tempering is computationally more expensive per step than the Gibbs sampler.

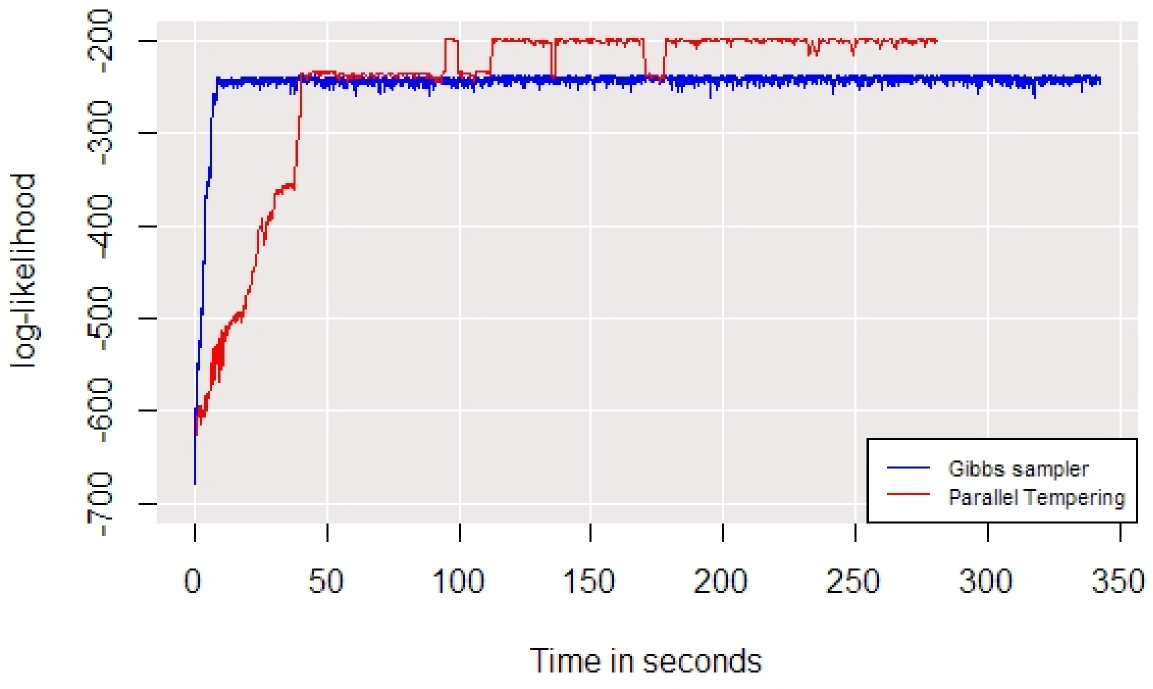

Figure 5.3.: Log-likelihood against time for the Gibbs sampler (red) and parallel tempering (blue).

We see that the Gibbs sampler initially achieves a higher log-likelihood as it finds a local optimum of faster than parallel tempering. However, it gets stuck there, while parallel tempering eventually discovers the true underlying feature matrix, giving a higher log-likelihood after around 100 seconds.

This shows that multimodality of the target distribution is an issue that needs to be addressed when conducting inference in latent feature models. Parallel tempering seems to be an appropriate tool, as it is designed to efficiently sample from multimodal target distributions. In this case it successfully recovered the true feature matrix **Z**, whereas the Gibbs sampler got stuck at a local optimum with one additional feature.

6. Conclusion and Discussion

Inspired by the lifting idea for designing non-reversible MCMC algorithms (see e.g. [4, 28, 45]) and based on the Generalized Metropolis-Hastings framework introduced in [23] we proposed non-reversible parallel tempering algorithms in order to tackle the diffusive random walk behavior the replicas exhibit in the temperature space in standard parallel tempering algorithms.

We introduced a lifting variable (n, ϵ), which indicates the temperature at which the replica is to be swapped and the direction, that is with which adjacent temperature it is to be swapped. In algorithm 3 this lifting variable moves around the temperature space as if the temperatures were assembled along a line. However, this introduces a baseline mode effect, as after a successful swap of the replicas at temperatures $N-1$ and N the lifting variable forces another swap with these two replicas, returning the old mode to temperature N. In algorithm 4 we allowed the lifting variable to jump between the highest and lowest temperature in order to mitigate this effect. In this case we can think of the lifting variable as moving around the temperature space as if it was a circle.

Indeed, the movement in the temperature space the non-reversible algorithms exhibit is not randomwalk-like, but show persistent movements in the same direction, allowing fast movement up and down the temperature space. However, if bottlenecks exist, replicas tend to spend a relatively long time between two bottlenecks.

Increasing the frequency with which swaps are attempted is expected to benefit mixing, see for example [34] and [35]. This can either be done by decreasing the number of Metropolis-Hastings steps between two swap attempts, which is trivial, or by proposing multiple swaps in each attempt. Simply introducing two copies of the lifting variable which operate independently of each other is suboptimal, since when both copies move together they undo each other's swaps. In algorithms 5 and 6 we show how the two copies of the lifting variable should interact in order to reduce the efficiency lost due to introducing multiple copies.

We compared our non-reversible algorithms with standard parallel tempering in a Gaussian mixture model and experimentally identified situations in which the non-reversible or reversible parallel tempering algorithms have a better performance. In different scenarios we confirmed the intuition on random walks given in section 2.1, that the non-reversible algorithms are expected to compare favorably with the standard algorithms when the number of temperatures is large relative to the total number of swaps attempted.

However, these are only experimental results rather than rigorous derivations of the differences between our non-reversible algorithms and standard tempering. We merely confirmed our intuitive expectations in a number of different scenarios, and further work would be necessary in order to show the general validity of our conclusion that the non-reversible algorithms have superior performance if the number of temperatures is large relative to the number of steps; one would also need to define what a reltively large number of temperatures means.

In bayesian nonparametric latent feature models the target distribution is typically on a high-dimensional space and multimodal. The classical Gibbs sampler used in [10], [13], [14] and [28] therefore has difficulties recovering the underlying feature matrix $\mathbf{Z}$. This is a natural situation to apply parallel tempering, and we demonstrated on an example using simulated data with a linear gaussian likelihood that the Gibbs sampler gets stuck in a local optimum, whereas parallel tempering eventually discovers the true underlying features. Many improvements have been suggested for the Gibbs sampler. The slice sampler in [38] will also suffer from the multimodal target distribution, whereas we intuitively expect the particle filter proposed in [46] to be able to better handle local optima, since the observations are introduced one by one, making it more unlikely to find a feature representation using more features than necessary. However, this is merely an intuition and requires further investigation. These and other extensions could be combined with parallel tempering in order to improve the sampling steps at fixed temperatures both in terms of mixing and computational cost.

Further work could include extending the non-reversible algorithms to the case with many copies of the lifting variable. Lingenheil et al. compared four different exchange schemes in [24] and alternatingly attempting to swap all even pairs of temperatures $(2i, 2i+1)$ and odd pairs $(2i-1, 2i)$ has the best performance in terms of the round trip rate of all schemes considered. While we could introduce that many copies of the lifting variable in algorithms 5 and 6, it is clear that this would undermine the idea that the lifting variable(s) can move up and down the temperature space fast. Other enhancements for parallel tempering could also be applied to our non-reversible algorithms. For instance, in [18] Katzgraber et al. propose a feedback algorithm, where each replica is labeled "up" or "down", depending on whether they last visited the highest or lowest temperature. For each temperature i it is counted how often up-replicas and how often down-replicas visit this temperature, and based on this the temperature set is iteratively updated in order to achieve a high round trip rate. In particular this targets bottlenecks, thus an optimized temperature set should greatly benefit our non-reversible algorithms, if there are no bottlenecks where the lifting variable could get stuck. Suwa and Todo proposed an algorithm reducing the rejection rate in MCMC algorithms in [36]. Itoh and Okumura applied this idea to parallel tempering and constructed an Replica Permutation algorithm. Further developments could include applying these ideas of reducing the rejection rate and replica permutations to our non-reversible algorithms, e.g. by only considering permutations which are possible in our non-reversible algorithms as opposed to all possible permutations.

References

[1] Atchadé, Y. F., Roberts, G. O. and Rosenthal, J. S. (2011). Towards optimal scaling of Metropolis-coupled Markov chain Monte Carlo. *Statistics and Computing*, **21**(4):555-568

[2] Bierkens, J. (2015). Non-reversible Metropolis-Hastings. *Statistics and Computing*, 1-16

[3] Bouchard-Côté, A., Vollmer, S. J. and Doucet, A.. The Bouncy Particle Sampler: A Non-Reversible Rejection-Free Markov Chain Monte Carlo Method. *Journal of the American Statistical Association, to appear. arXiv preprint arXiv:1510.02451.*

[4] Chen, F., Lovász, L. and Pak, I. (1999). Lifting Markov CHains to Speed up Mixing. In *Proceedings of the ACM symposium on Theory of Computing*, 275-281

[5] Diaconis, P., Holmes, S. and Neal, R. M. (2000). Analysis of a nonreversible Markov chain sampler. *The Annals of Applied Probability*, **10**, 726-752

[6] Earl, D. J. and Deem, M. W. (2005). Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7:3910-3916

[7] Fernandes, H. C. M. and Weigel, M. (2011). Non-reversible Monte Carlo simulations of spin models. *Computer Physics Communications* **182**, 1856-1859

[8] Gelman, A. and Shalizi, C. R. (2011). Philosophy and the practice of Bayesian statistics. In *The Oxford Handbook of Philosophy of Social Science.* Oxford University Press

[9] Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. *Computing Science and Statistics: Proceedings of 23rd Symposium on the Interface*, 156–163

[10] Ghahramani, Z., Griffiths, T. L. and Sollich, P. (2006). Bayesian nonparametric latent feature models. *Bayesian Statistics* **8**

[11] Görür, D., Jäkel, F. and Rasmussen, C. E. (2006). A Choice Model with Infinitely Many Latent Features. In *Proceedings of the 23rd International Conference on Machine Learning,* 361-368, ACM Press, New York

[12] Griffiths, T. L. and Ghahramani, Z. (2005). Infinite Latent Feature Models and the Indian Buffet Process. Technical Report 2005-001, Gatsby Computational Neuroscience Unit

[13] Griffiths, T. L. and Ghahramani, Z. (2006). Infinite Latent Feature Models and the Indian Buffet Process. In *Advances in Neural Processing Systems 18.* MIT Press, Cambridge, MA

[14] Griffiths, T. L. and Ghahramani, Z. (2011). The Indian Buffet Process: An Introduction and Review. *Journal of Machine Learning Research* **12**, 1185-1224

[15] Hasenbusch, M. and Schaefer, S. (2010). Speeding up parallel tempering simulations. *Phys. Rev.* E 82, 046707

[16] Hjort, N. L. (1990). Nonparametric Bayes estimators based on beta processes in models for life history data. *The Annals of Statistics*, 18(3):1259–1294

[17] Itoh, S. G. and Okumura, H. (2012). Replica-Permutation Method with the Suwa-Todo Algorithm beyond the Replica-Exchange Method. *Journal of Chemical Theory and Computation* 9(1), 570-581

[18] Katzgraber, H. G., Trebst, S., Huse, D. A. and Troyer, M. (2006). Feedback-optimized parallel tempering Monte Carlo. *Journal of Statistical Mechanics* P03018

[19] Klartag, B. (2006). A Central Limit Theorem for Convex Sets. *Invent. Math.* 168, 91–131

[20] Kofke, D. A. (2002). On the acceptance probability of replica-exchange Monte Carlo trials. *Journal of Chemical Physics* **117**, 10852-10852

[21] Kone, A. and Kofke, D. A. (2005). Selection of temperature intervals for parallel-tempering simulations. *Journal of Chemical Physics* **122**, 206101

[22] Lee, A., Yau, C., Giles, M. B., Doucet, A. and Holmes, C. C. (2010). On the Utility of Graphics Cards to Perform Massively Parallel Simulation of Advanced Monte Carloe Methods. *Journal of Computational and Graphical Statistics*, 1-21

[23] Lelievre, T., Rousset, M. and Stoltz, G. (2010). *Free Energy Computation: A Mathematical Perspective.* Imperial College Press, London

[24] Lingenheil, M., Denschlag, R., Mathias, G. and Tavan, P. (2009). Efficiency of exchange schemes in replica exchange. *Chemical Physics Letters* **478** 80-84

[25] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics* **21**(6):1087-1092

[26] Miasojedow, B., Moulines, E. and Vihola, M. (2013). An adaptive parallel tempering algorithm. *Journal of Computational and Graphical Statistics* **22**, 649-664

[27] Michel, M. (2016). *Irreversible Markov chains by the factorized Metropolis filter: Algorithms and applications in particle systems and spin models.* PhD thesis, Ecole Normale Supereure de Paris

[28] Miller, K. T. (2011). *Bayesian Nonparametric Latent Feature Models.* PhD thesis, UC Berkely

[29] Neal, R. M. (2000). Markov Chain Sampling Methods for Dirichlet Process Mixture Models. *Journal of Computational and Graphical Statistics,* 9:249-265

[30] Paquet, U. (2007). *Bayesian Inference for Latent Variable Models.* PhD thesis, University of Cambridge

[31] Predescu, C., Predescu, M. and Ciobanu, C. (2004). The imcomplete beta function law for parallel tempering sampling of classical canonical systems. *Journal of Chemical Physics* **120**, 4119

[32] Rasmussen, C. E. and Williams, C. (2006). *Gaussian Processes for Machine Learning.* MIT Press

[33] Rathore, N., Chopra, M. and de Pablo, J. J. (2005). Optimal allocation of replicas in parallel tempering simulations. *Journal of Chemical Physics* **122**, 024111

[34] Sindhikara, D., Emerson, D. J. and Roitberg, A. E. (2010). Exchange Often and Properly in Replica Exchange Molecular Dynamics. *Journal of Chemical Theory and Computation* 6(9), 2804-2808

[35] Sindhikara, D., Meng, Y. and Roitberg, A. E. (2008). Exchange frequency in replica exchange molecular dynamics. *Journal of Chemical Physics* **128**, 024103

[36] Suwa, H. and Todo, S. (2010). Markov chain Monte Carlo method without detailed balance. *Physical Review Letters* **105**, 120603

[37] Swendsen, R. H. and Wang, J. S. (1986). Replica Monte Carlo Simulation of Spin-Glasses. *Physical Review Letters* **57**, 2607-2609

[38] Teh, Y. W., Görür, D. and Ghahramani, Z. (2007). Stick-breaking Construction for the Indian Buffet Process. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*

[39] Teh, Y. W., Jordan, M. I., Beal, M. J. and Blei, D. M. (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association,* 101(476):1566-1581

[40] Teh, Y. W. (2010). Dirichlet Processes. In *Encyclopedia of Machine Learning.* Springer

[41] Thibaux, R. and Jordan, M. I. (2007). Hierarchical Beta Processes and the Indian Buffet Process. In *Proceedings of the Conference on Artificial Intelligence and Statistics*

[42] Titsias, M. (2008). The Infinite Gamma-Poisson Feature Model. In *Advances in Neural Information Processing Systems 20.* MIT Press, Cambridge, MA

[43] Turitsyn, K. S., Chertkov, M. and Vucelja, M. (2011). Irreversible Monte Carlo Algorithms for Efficient Sampling. *Physica D* **240**, 410-414

[44] Doshi-Velez, F. and Ghahramani, Z. (2009). Accelerated sampling for the Indian Buffet Process. In *Proceeding sof the International Conference on Machine learning*

[45] Vucelja, M. (2016). Lifting - A nonreversible Markov chain Monte Carlo algorithm. *American Journal of Physics* **84**(12), 958-968

[46] Wood, F. and Griffiths, T. L. (2006). Particle Filtering for Nonparametric Bayesian Matrix Factorization. In *Advances in Neural Information Processing Systems*

Appendices

A. Invariant distribution of Algorithms 3-6

We have to ensure that all assumptions of the Generalized Metropolis-Hastings framework are satisfied.

From the definition of the function S for Algorithms 4-7 it is immediately clear that it is an involution. Since S only modifies the auxiliary variable ϵ and ϵ_1, ϵ_2 respectively, for which we choose the uniform distribution on their defined state space $\{-1, 1\}$, S is also measure preserving.

For each algorithm we are using a deterministig proposal of the form

$$Q((x, v), (dx', dv')) = \delta_{\Phi(x,v)}(dx', dv').$$

As was shown in section 2.2 it is sufficient to show that $\Phi^{-1} = S \circ \Phi \circ S$ holds in this case; then the Markov chain with the transition kernel defined by the involution S and the proposal Q has the correct invariant distribution.

A.1. Algorithm 3

<u>Case 1:</u> $1 < n, n - \epsilon < N$

$$\Phi^{-1}(x, n, \epsilon) = (x_{(n,n-\epsilon)}, n - \epsilon, \epsilon)$$
$$S \circ \Phi \circ S(x, n, \epsilon) = S \circ \Phi(x, n, -\epsilon) = S(x_{(n,n-\epsilon)}, n - \epsilon, -\epsilon) = (x_{(n,n-\epsilon)}, n - \epsilon, \epsilon)$$

<u>Case 2:</u> $n = 1$ (then $\epsilon = 1$) or $n = N$ (then $\epsilon = -1$)

$$\Phi^{-1}(x, n, \epsilon) = (x_{(n,n+\epsilon)}, n + \epsilon, -\epsilon)$$
$$S \circ \Phi \circ S(x, n, \epsilon) = S \circ \Phi(x, n, \epsilon) = S(x_{(n,n+\epsilon)}, n + \epsilon, \epsilon) = (x_{(n,n+\epsilon)}, n + \epsilon, -\epsilon)$$

<u>Case 3:</u> $n - \epsilon \in \{1, N\}$

$$\Phi^{-1}(x, n, \epsilon) = (x_{(n,n-\epsilon)}, n - \epsilon, \epsilon)$$
$$S \circ \Phi \circ S(x, n, \epsilon) = S \circ \Phi(x, n, -\epsilon) = S(x_{(n,n-\epsilon)}, n - \epsilon, \epsilon) = (x_{(n,n-\epsilon)}, n - \epsilon, -\epsilon)$$

A.2. Algorithm 4

Case 1: not $(n,\epsilon)=(1,1)$ or $(n,\epsilon)=(N-1,-1)$

$$\begin{gathered}\Phi^{-1}(x,n,\epsilon)=(x_{(n,n-\epsilon)},n-\epsilon,\epsilon)\\ S\circ\Phi\circ S(x,n,\epsilon)=S\circ\Phi(x,n,-\epsilon)=S(x_{(n,n-\epsilon)},n-\epsilon,-\epsilon)=(x_{(n,n-\epsilon)},n-\epsilon,\epsilon)\end{gathered}$$

Case 2: $(n,\epsilon)=(N-1,-1)$

$$\begin{gathered}\Phi^{-1}(x,N-1,-1)=(x_{(N-1,N)},1,-1)\\ S\circ\Phi\circ S(x,N-1,-1)=S\circ\Phi(x,N-1,1)=S(x_{(N-1,N)},1,1)=(x_{(N-1,N)},1,-1)\end{gathered}$$

Case 3: $(n,\epsilon)=(1,1)$

$$\begin{gathered}\Phi^{-1}(x,1,1)=(x_{(N-1,N)},N-1,1)\\ S\circ\Phi\circ S(x,1,1)=S\circ\Phi(x,1,-1)=S(x_{(N-1,N)},N-1,-1)=(x_{(N-1,N)},N-1,1)\end{gathered}$$

A.3. Algorithm 5

Case 1: $n_1\neq 1$, $n_1-\epsilon_1\neq n_2$ and $n_1-\epsilon_1\neq 1$

$$\begin{aligned}\Phi^{-1}(x,n_1,\epsilon_1,n_2,\epsilon_2)&=(x_{(n_1,n_1-\epsilon_1)},n_1-\epsilon_1,\epsilon_1,n_2,\epsilon_2)\\ S\circ\Phi\circ S(x,n_1,\epsilon_1,n_2,\epsilon_2)&=S\circ\Phi(x,n_1,-\epsilon_1,n_2,\epsilon_2)=S(x_{(n_1,n_1-\epsilon_1)},n_1-\epsilon_1,-\epsilon_1,n_2,\epsilon_2)\\ &=(x_{(n_1,n_1-\epsilon_1)},n_1-\epsilon_1,\epsilon_1,n_2,\epsilon_2)\end{aligned}$$

Case 2: $n_1\neq 1$ and $n_1-\epsilon_1=1$ (in particular $n_1-\epsilon_1\neq n_2$)

$$\begin{aligned}\Phi^{-1}(x,n_1,\epsilon_1,n_2,\epsilon_2)&=(x_{(n_1,n_1-\epsilon_1)},n_1-\epsilon_1,\epsilon_1,n_2,\epsilon_2)\\ S\circ\Phi\circ S(x,n_1,\epsilon_1,n_2,\epsilon_2)&=S\circ\Phi(x,n_1,-\epsilon_1,n_2,\epsilon_2)=S(x_{(n_1,n_1-\epsilon_1)},n_1-\epsilon_1,\epsilon_1,n_2,\epsilon_2)\\ &=(x_{(n_1,n_1-\epsilon_1)},n_1-\epsilon_1,\epsilon_1,n_2,\epsilon_2)\end{aligned}$$

Case 3: $n_1=1$ (then $\epsilon_1=1$)and $n_2\neq 2$

$$\begin{aligned}\Phi^{-1}(x,n_1,\epsilon_1,n_2,\epsilon_2)&=(x_{(n_1,n_1+\epsilon_1)},n_1+\epsilon_1,-\epsilon_1,n_2,\epsilon_2)\\ S\circ\Phi\circ S(x,n_1,\epsilon_1,n_2,\epsilon_2)&=S\circ\Phi(x,n_1,\epsilon_1,n_2,\epsilon_2)=S(x_{(n_1,n_1+\epsilon_1)},n_1+\epsilon_1,\epsilon_1,n_2,\epsilon_2)\\ &=(x_{(n_1,n_1+\epsilon_1)},n_1+\epsilon_1,-\epsilon_1,n_2,\epsilon_2)\end{aligned}$$

Case 4: $n_1=1$ (then $\epsilon_1=1$)and $n_2=2$

$$\begin{aligned}\Phi^{-1}(x,n_1,\epsilon_1,n_2,\epsilon_2)&=(x_{(n_1,n_1+\epsilon_1)},n_1,\epsilon_1,n_2,\epsilon_2)\\ S\circ\Phi\circ S(x,n_1,\epsilon_1,n_2,\epsilon_2)&=S\circ\Phi(x,n_1,\epsilon_1,n_2,\epsilon_2)=S(x_{(n_1,n_1+\epsilon_1)},n_1,\epsilon_1,n_2,\epsilon_2)\\ &=(x_{(n_1,n_1+\epsilon_1)},n_1,\epsilon_1,n_2,\epsilon_2)\end{aligned}$$

<u>Case 5:</u> $1 \neq n_1 = n_2 - 1$, $\epsilon_1 = -1$ and $n_2 \neq N$

$$\begin{aligned}\Phi^{-1}(x, n_1, \epsilon_1, n_2, \epsilon_2) &= (x_{(n_1,n_2)}, n_1, -\epsilon_1, n_2, -\epsilon_2)\\ S \circ \Phi \circ S(x, n_1, \epsilon_1, n_2, \epsilon_2) &= S \circ \Phi(x, n_1, -\epsilon_1, n_2, \epsilon_2) = S(x_{(n_1,n_2)}, n_1, \epsilon_1, n_2, -\epsilon_2)\\ &= (x_{(n_1,n_2)}, n_1, -\epsilon_1, n_2, -\epsilon_2)\end{aligned}$$

<u>Case 6:</u> $1 \neq n_1 = n_2 - 1$, $\epsilon_1 = -1$ and $n_2 = N$

$$\begin{aligned}\Phi^{-1}(x, n_1, \epsilon_1, n_2, \epsilon_2) &= (x_{(n_1,n_2)}, n_1, -\epsilon_1, n_2, \epsilon_2)\\ S \circ \Phi \circ S(x, n_1, \epsilon_1, n_2, \epsilon_2) &= S \circ \Phi(x, n_1, -\epsilon_1, n_2, \epsilon_2) = S(x_{(n_1,n_2)}, n_1, \epsilon_1, n_2, \epsilon_2)\\ &= (x_{(n_1,n_2)}, n_1, -\epsilon_1, n_2, \epsilon_2)\end{aligned}$$

A.4. Algorithm 6

<u>Case 1:</u> $n_1 - \epsilon_1 \neq n_2$ and $(n_1, \epsilon_1) \neq (1, 1), (N - 1, -1)$

$$\begin{aligned}\Phi^{-1}(x, n_1, \epsilon_1, n_2, \epsilon_2) &= (x_{(n_1,n_1-\epsilon_1)}, n_1 - \epsilon_1, \epsilon_1, n_2, \epsilon_2)\\ S \circ \Phi \circ S(x, n_1, \epsilon_1, n_2, \epsilon_2) &= S \circ \Phi(x, n_1, -\epsilon_1, n_2, \epsilon_2) = S(x_{(n_1,n_1-\epsilon_1)}, n_1 - \epsilon_1, -\epsilon_1, n_2, \epsilon_2)\\ &= (x_{(n_1,n_1-\epsilon_1)}, n_1 - \epsilon_1, \epsilon_1, n_2, \epsilon_2)\end{aligned}$$

<u>Case 2:</u> $n_1 - \epsilon_1 = n_2$ and $(n_1, \epsilon_1) \neq (1, 1), (N - 1, -1)$

$$\begin{aligned}\Phi^{-1}(x, n_1, \epsilon_1, n_2, \epsilon_2) &= (x_{(n_1,n_2)}, n_1, -\epsilon_1, n_2, -\epsilon_2)\\ S \circ \Phi \circ S(x, n_1, \epsilon_1, n_2, \epsilon_2) &= S \circ \Phi(x, n_1, -\epsilon_1, n_2, \epsilon_2) = S(x_{(n_1,n_2)}, n_1, \epsilon_1, n_2, -\epsilon_2)\\ &= (x_{(n_1,n_2)}, n_1 - \epsilon_1, -\epsilon_1, n_2, -\epsilon_2)\end{aligned}$$

<u>Case 3:</u> $(n_1, \epsilon_1) = (1, 1)$ and $n_2 \neq N - 1$

$$\begin{aligned}\Phi^{-1}(x, 1, 1, n_2, \epsilon_2) &= (x_{(N-1,N)}, N - 1, 1, n_2, \epsilon_2)\\ S \circ \Phi \circ S(x, 1, 1, n_2, \epsilon_2) &= S \circ \Phi(x, 1, -1, n_2, \epsilon_2) = S(x_{(N-1,N)}, N - 1, -1, n_2, \epsilon_2)\\ &= (x_{(N-1,N)}, N - 1, 1, n_2, \epsilon_2)\end{aligned}$$

<u>Case 4:</u> $(n_1, \epsilon_1) = (1, 1)$ and $n_2 = N - 1$

$$\begin{aligned}\Phi^{-1}(x, 1, 1, N - 1, \epsilon_2) &= (x_{(N-1,N)}, 1, -1, N - 1, -\epsilon_2)\\ S \circ \Phi \circ S(x, 1, 1, N - 1, \epsilon_2) &= S \circ \Phi(x, 1, -1, N - 1, \epsilon_2) = S(x_{(N-1,N)}, 1, 1, N - 1, -\epsilon_2)\\ &= (x_{(N-1,N)}, 1, -1, N - 1, -\epsilon_2)\end{aligned}$$

<u>Case 5:</u> $(n_1, \epsilon_1) = (N - 1, -1)$ and $n_2 \neq 1$

$$\begin{aligned}\Phi^{-1}(x, N - 1, -1, n_2, \epsilon_2) &= (x_{(N-1,N)}, 1, -1, n_2, \epsilon_2)\\ S \circ \Phi \circ S(x, N - 1, -1, n_2, \epsilon_2) &= S \circ \Phi(x, N - 1, 1, n_2, \epsilon_2) = S(x_{(N-1,N)}, 1, 1, n_2, \epsilon_2)\\ &= (x_{(N-1,N)}, 1, -1, n_2, \epsilon_2)\end{aligned}$$

Case 6: $(n_1, \epsilon_1) = (N-1, -11)$ and $n_2 = 1$

$$\begin{aligned}
\Phi^{-1}(x, N-1, -1, 1, \epsilon_2) &= (x_{(N-1,N)}, N-1, 1, 1, -\epsilon_2) \\
S \circ \Phi \circ S(x, N-1, -1, 1, \epsilon_2) &= S \circ \Phi(x, N-1, 1, 1, \epsilon_2) = S(x_{(N-1,N)}, N-1, -1, 1, -\epsilon_2) \\
&= (x_{(N-1,N)}, N-1, 1, 1, -\epsilon_2)
\end{aligned}$$

B. Diagnostic Plots

B.1. Diagnostic plots for Scenario 1

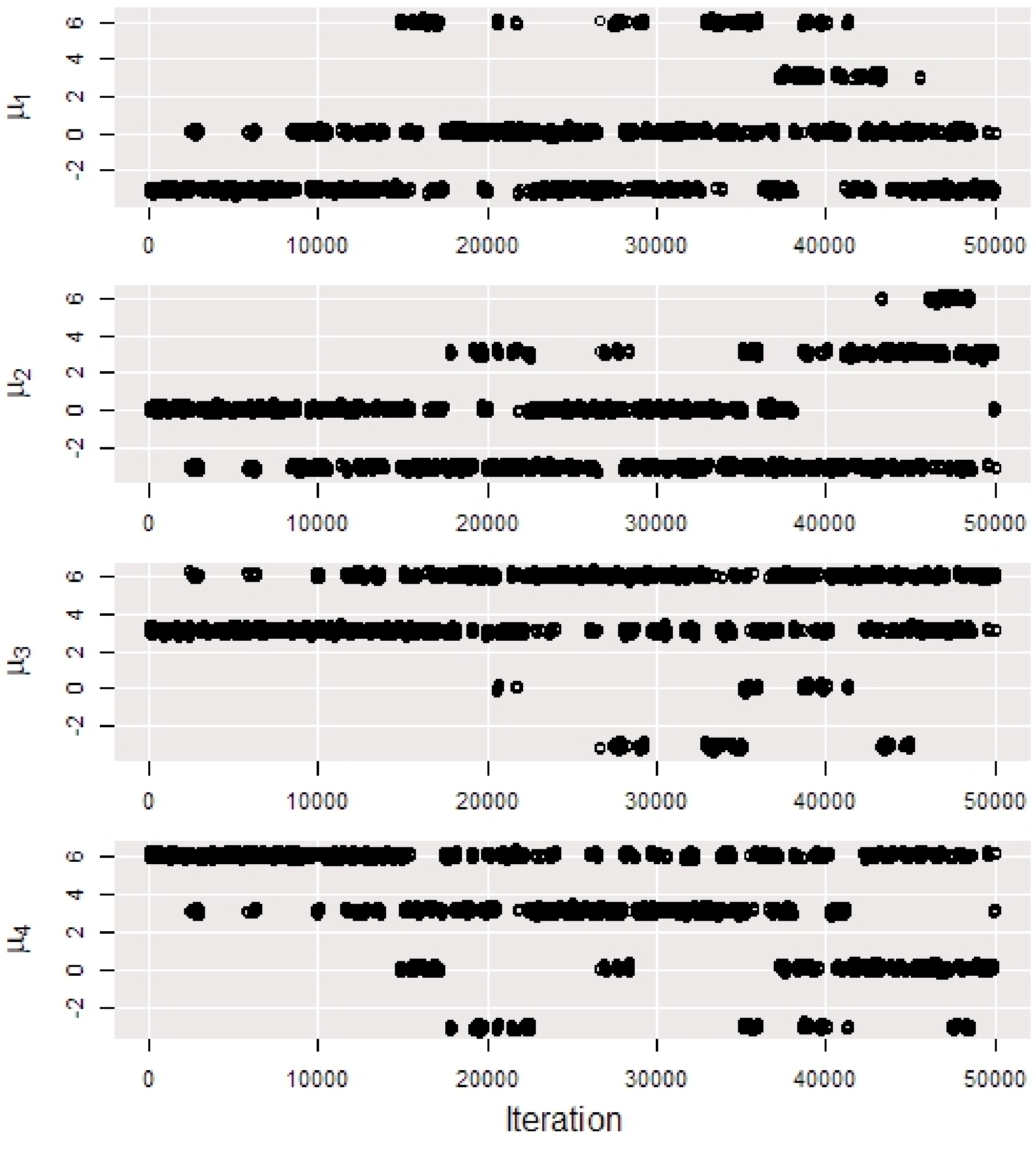

Figure B.1.: Traceplots of the estimates of the means of the four mixture components in PT. We see reasonable mixing.

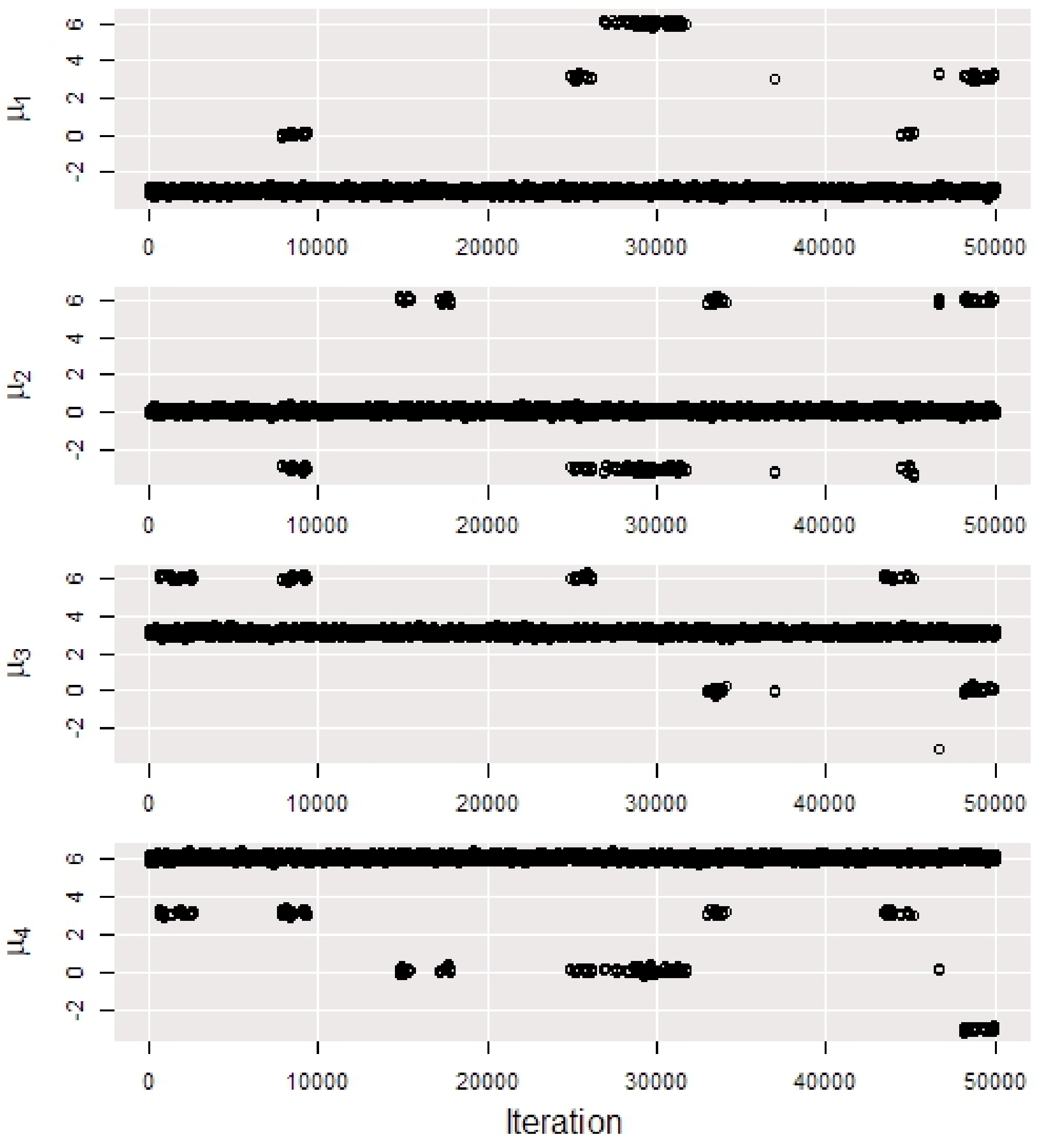

Figure B.2.: Traceplots of the estimates of the means of the four mixture components in LPT. While new modes are frequently found, it always returns to the "baseline mode" the Markov chain started in, $(-3, 0, 3, 6)$.

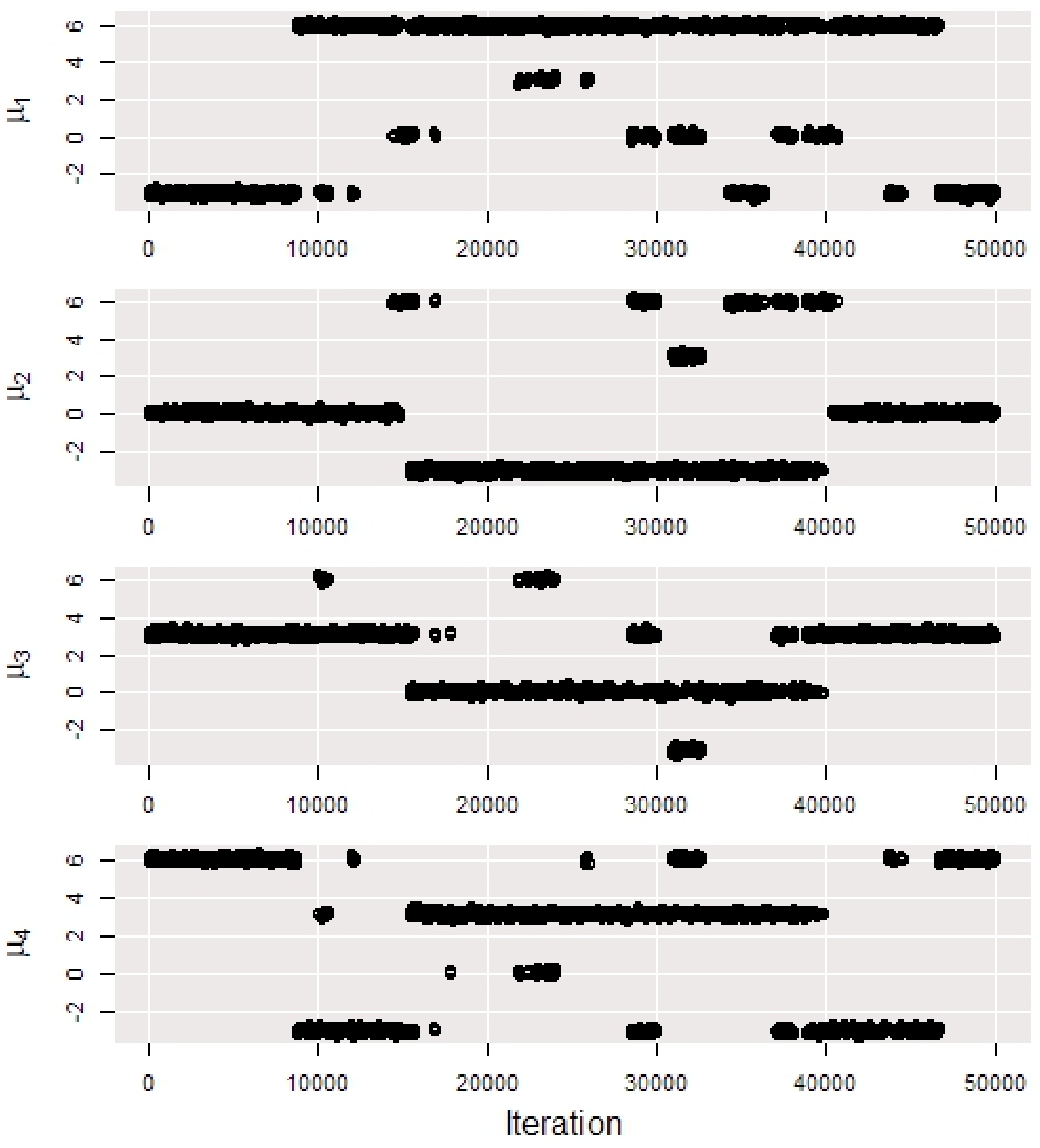

Figure B.3.: Traceplots of the estimates of the means of the four mixture components in LPTC. Allowing the current state to jump between 1 and $N-1$ indeed resolved the baseline mode issue.

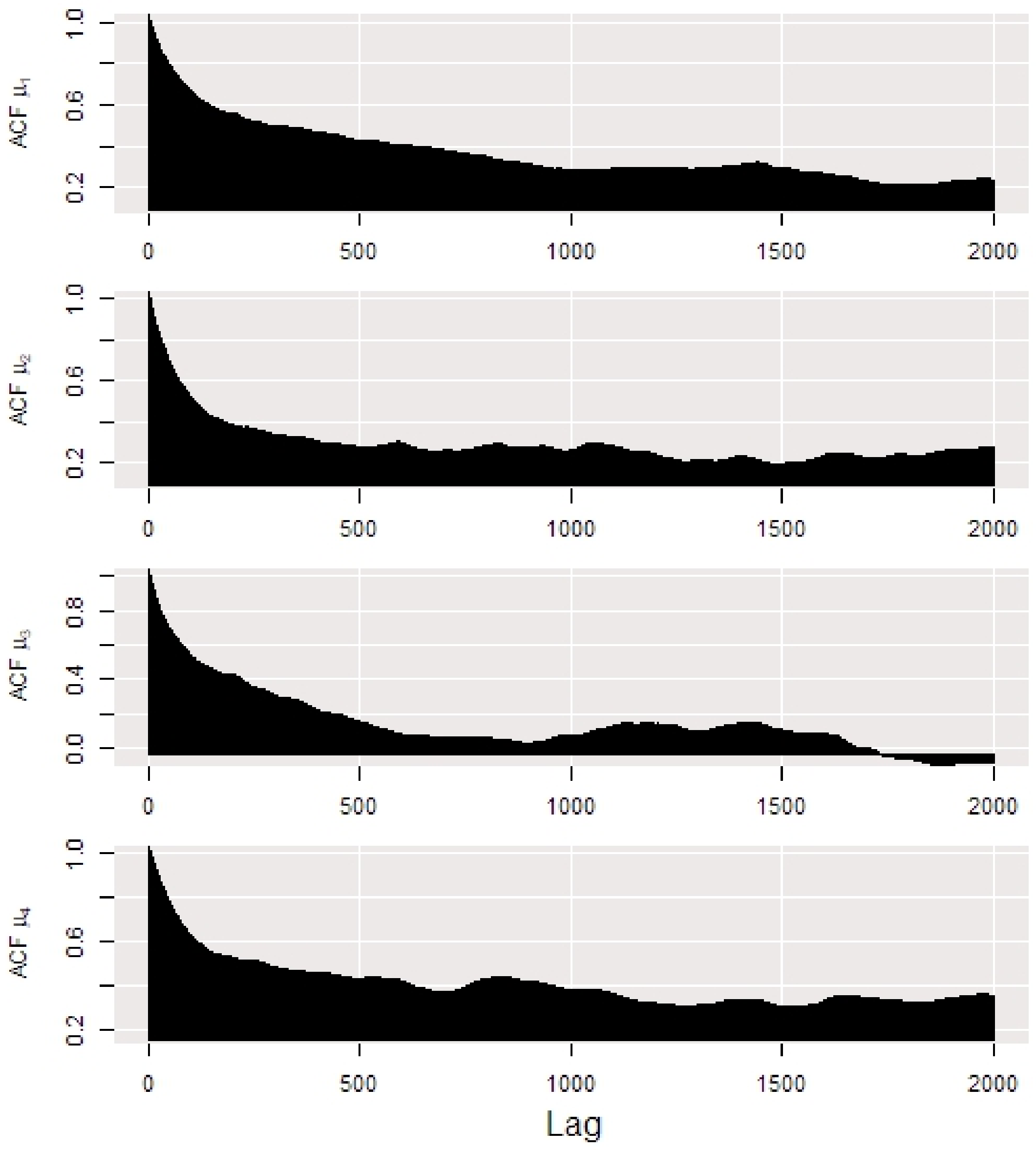

Figure B.4.: Autocorrelation for each of the four mixture components in PT. The autocorrelation falls off quickly compared to LPTC, as PT moves more freely between the modes.

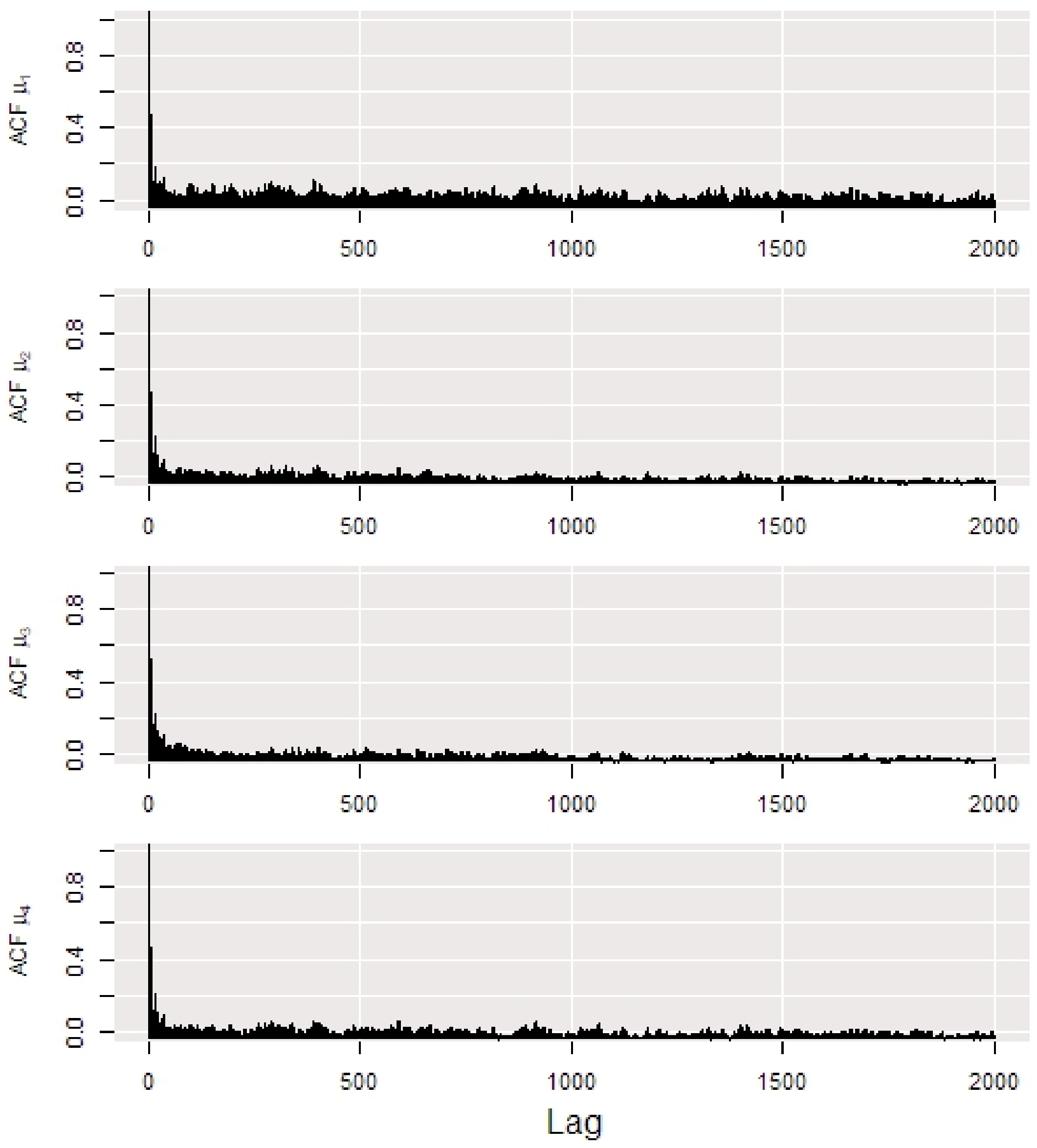

Figure B.5.: Traceplots of the estimates of the means of the four mixture components in LPT. This figure is highly misleading, since most time is is spent in one mode. Therefore, the ACF describes that the autocorrelation is low locally within that one mode.

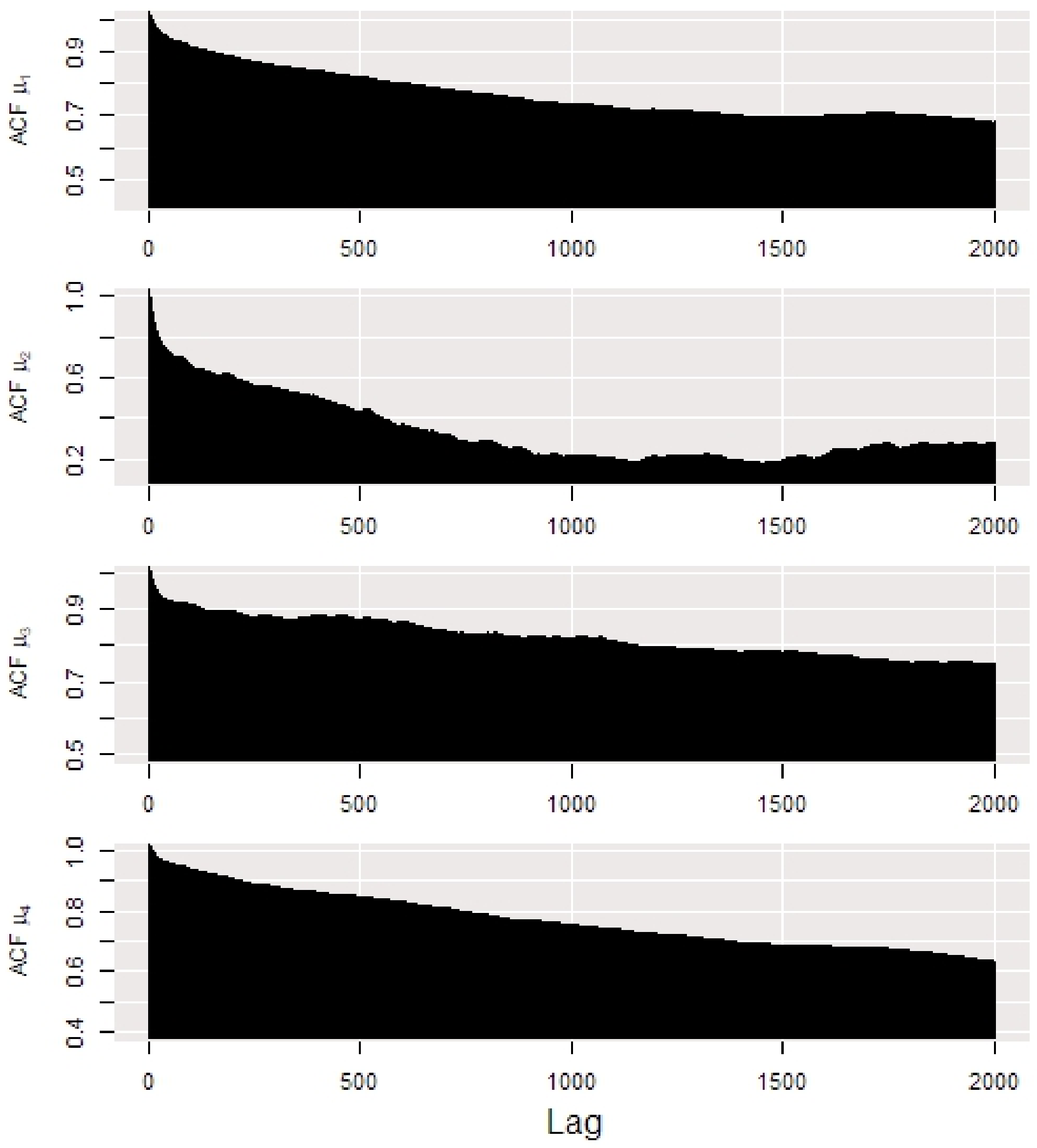

Figure B.6.: Traceplots of the estimates of the means of the four mixture components in LPTC. The autocorrelation function falls off slowly, since LPTC does spend more time in different modes than LPT, but cannot move between the modes as freely as PT.

B.2. Diagnostic plots for Scenario 2

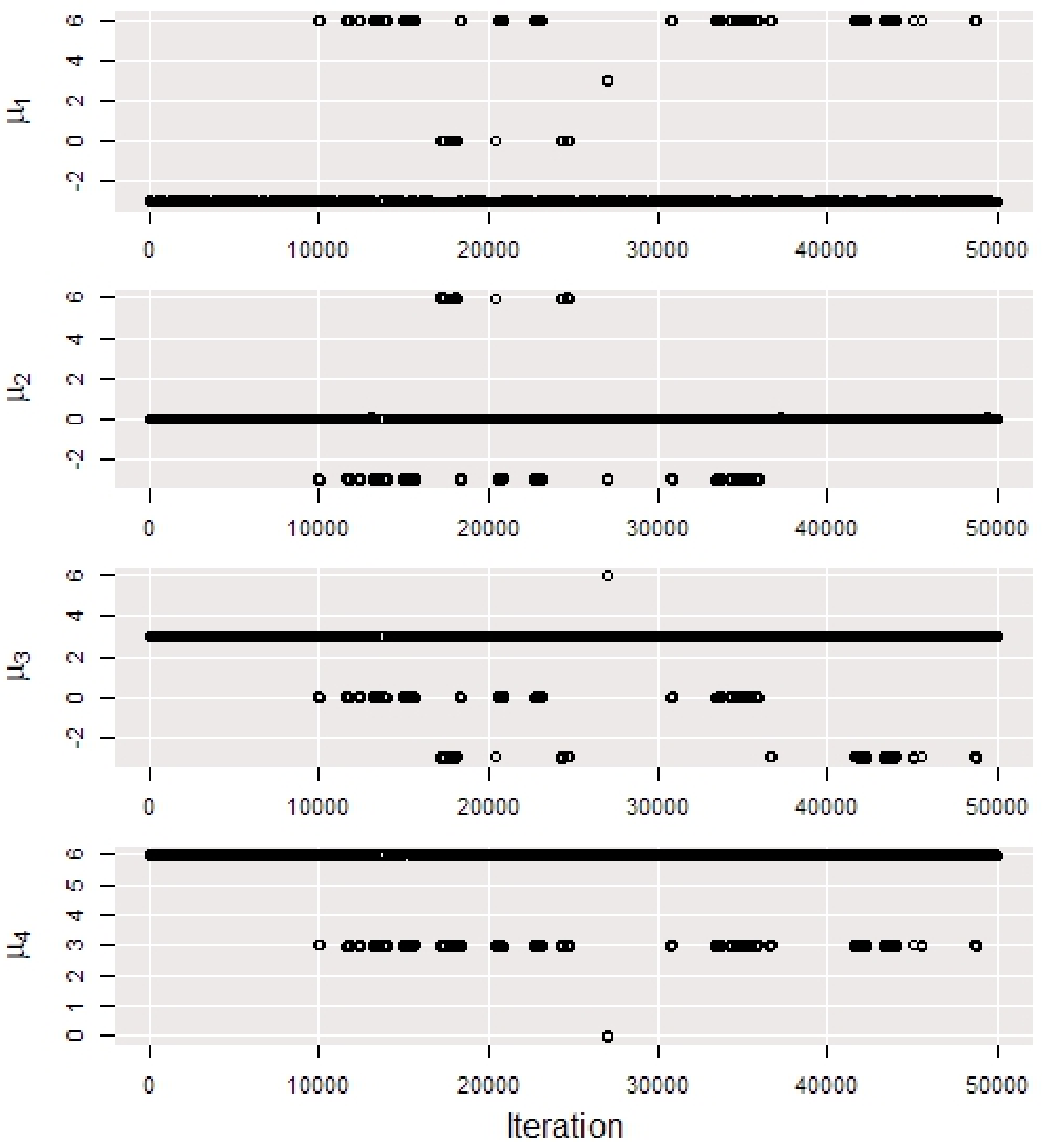

Figure B.7.: Traceplots of the estimates of the means of the four mixture components in PT2. We see that most time is spent in one mode, since around temperatures 20-40 only 4 replicas are in different modes and the rest are all in the same initial mode.

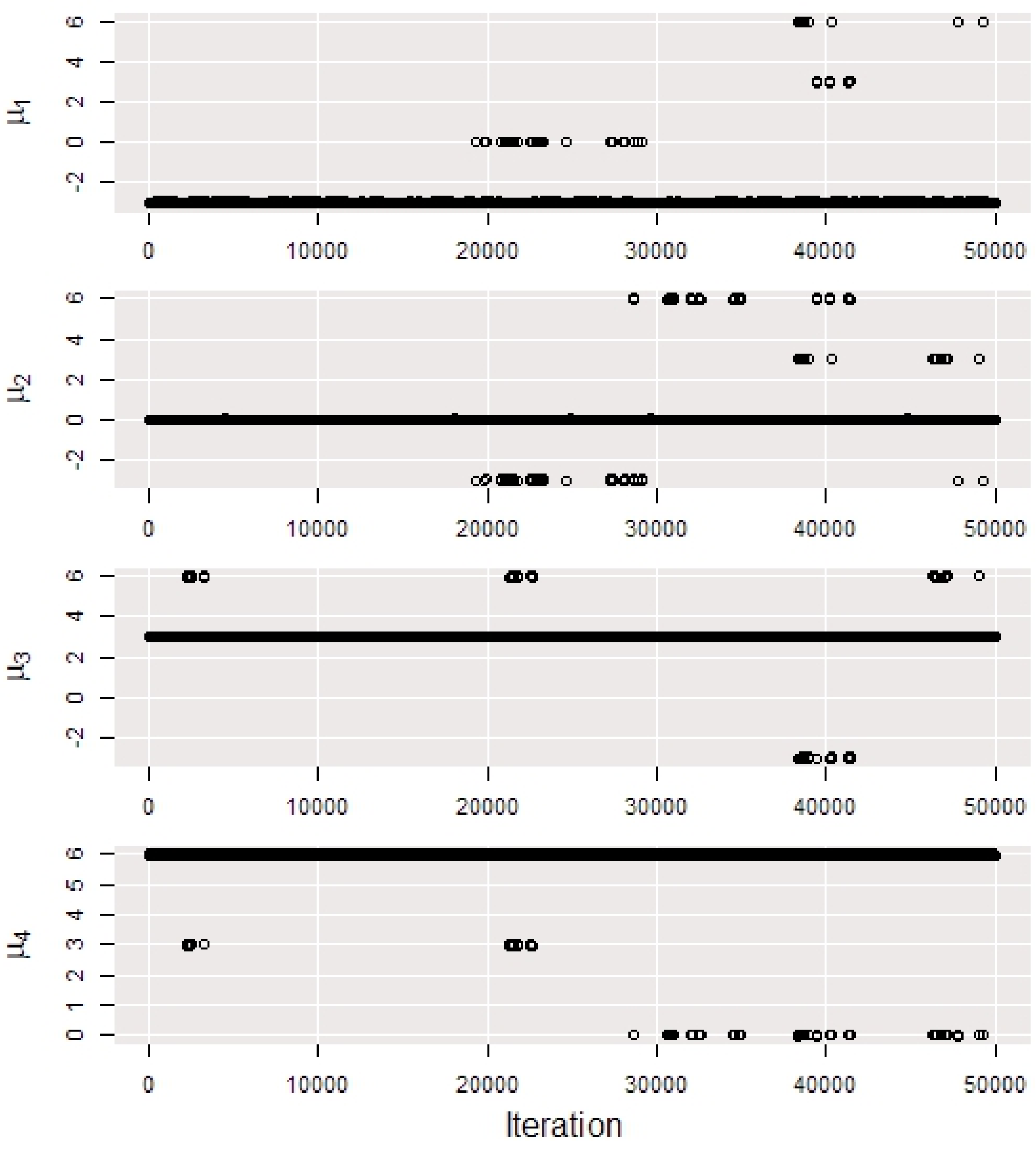

Figure B.8.: Traceplots of the estimates of the means of the four mixture components in LPT2. As above we observe the baseline mode behaviour.

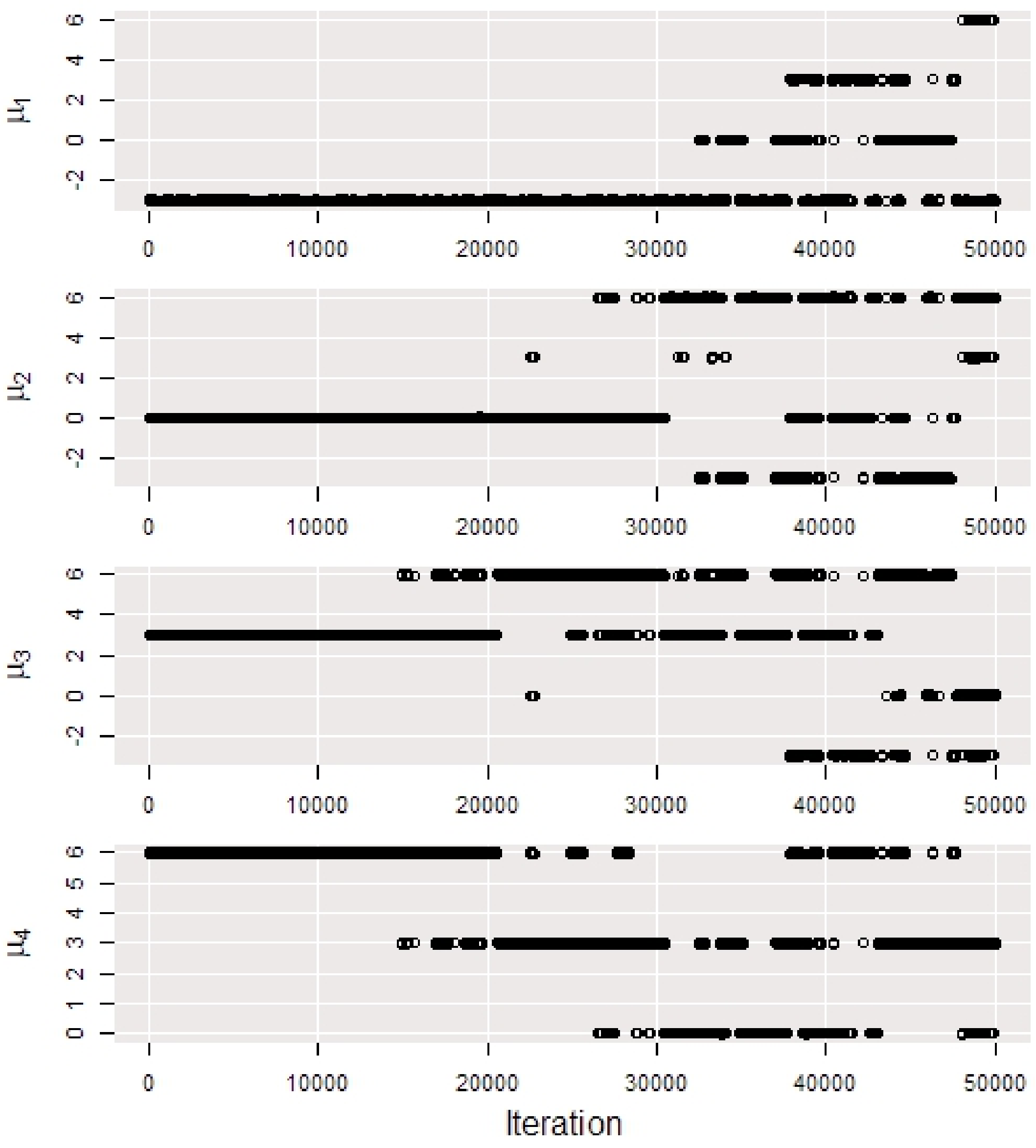

Figure B.9.: Traceplots of the estimates of the means of the four mixture components in LPTC2. Allowing the current state to jump between 1 and $N-1$ indeed resolved the baseline mode issue.

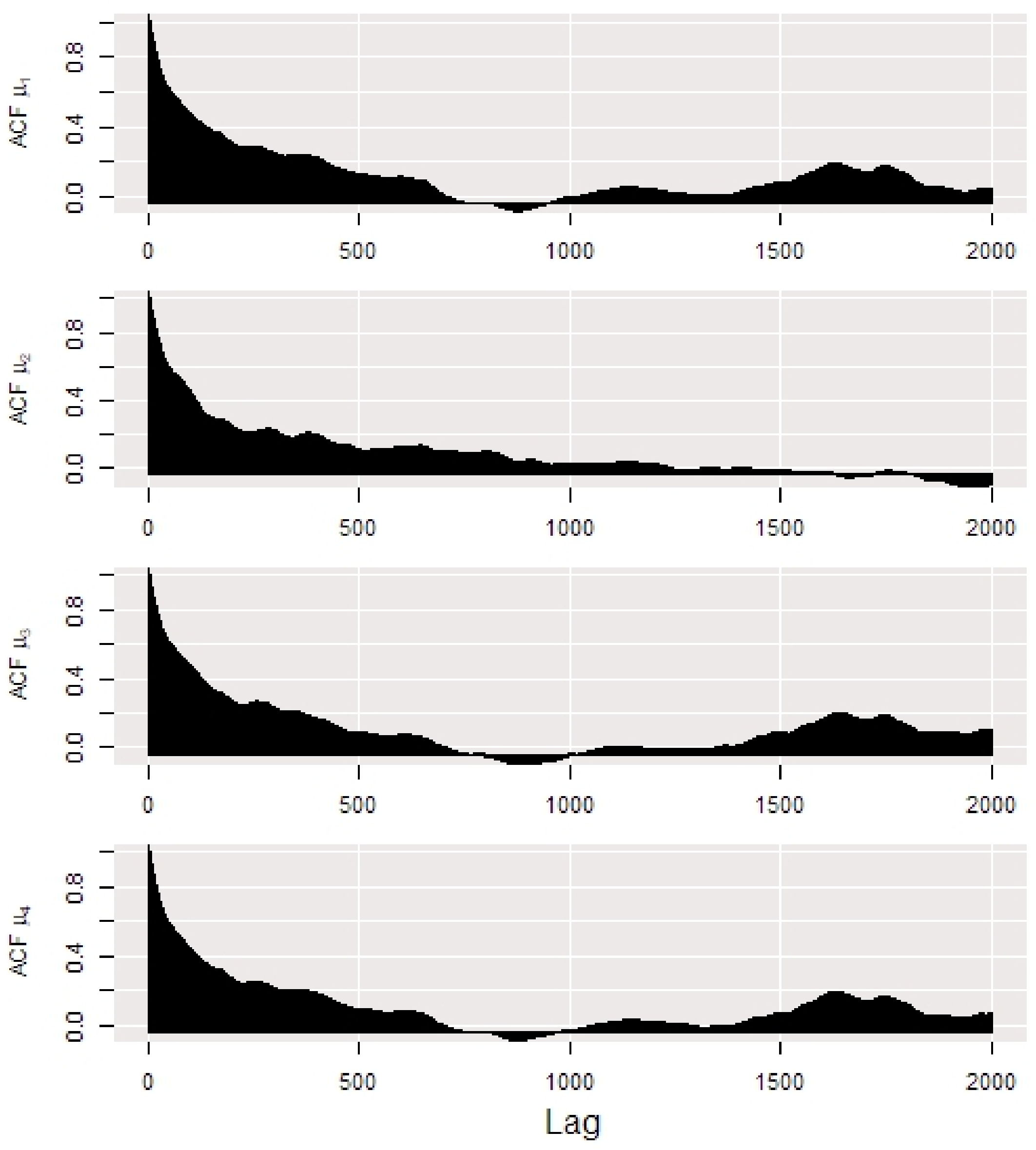

Figure B.10.: Autocorrelation for each of the four mixture components in PT. The autocorrelation falls off misleadingly fast since most time is spent in one mode.

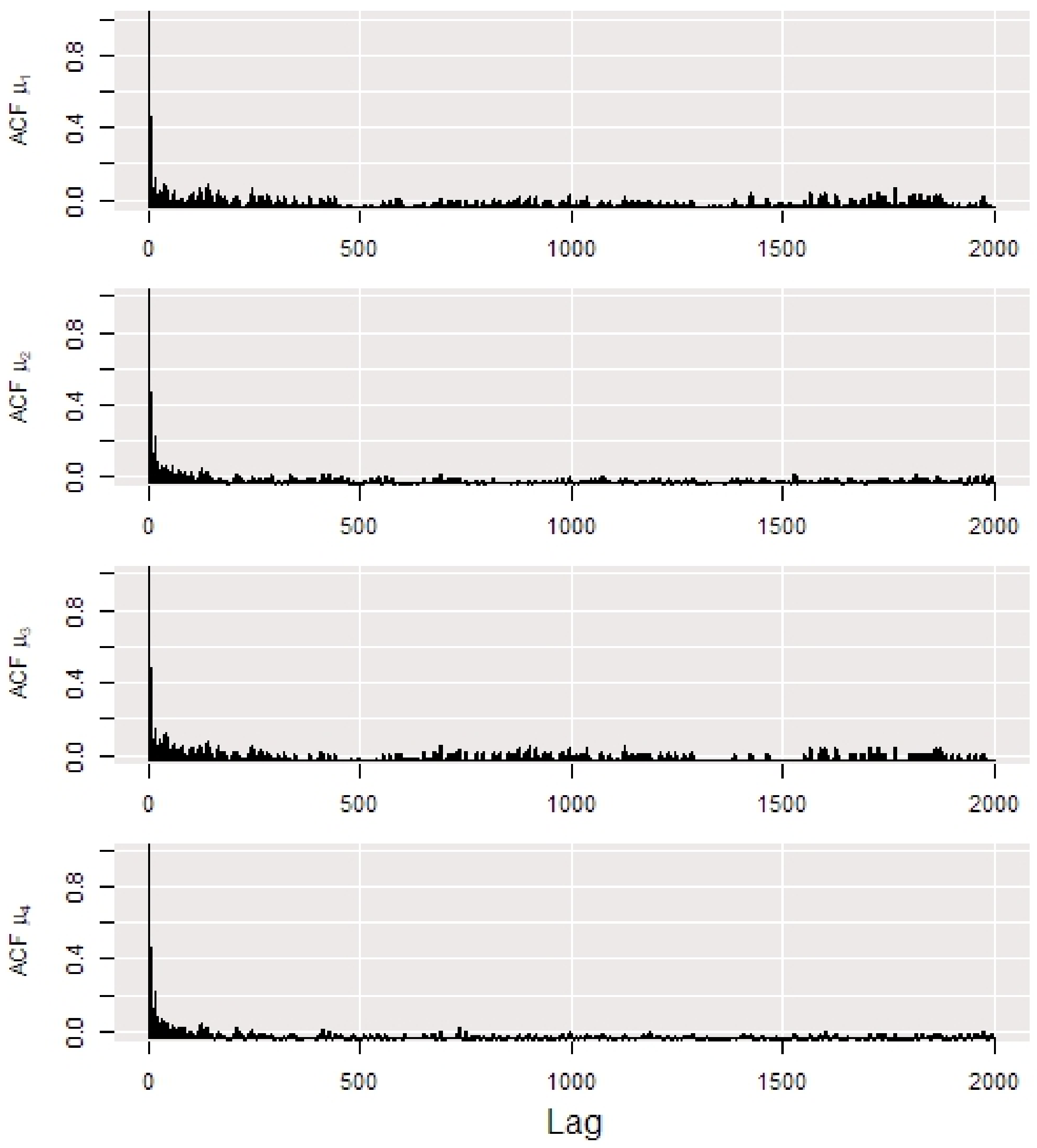

Figure B.11.: Traceplots of the estimates of the means of the four mixture components in LPT2. This figure is highly misleading, since most time is is spent in one mode.

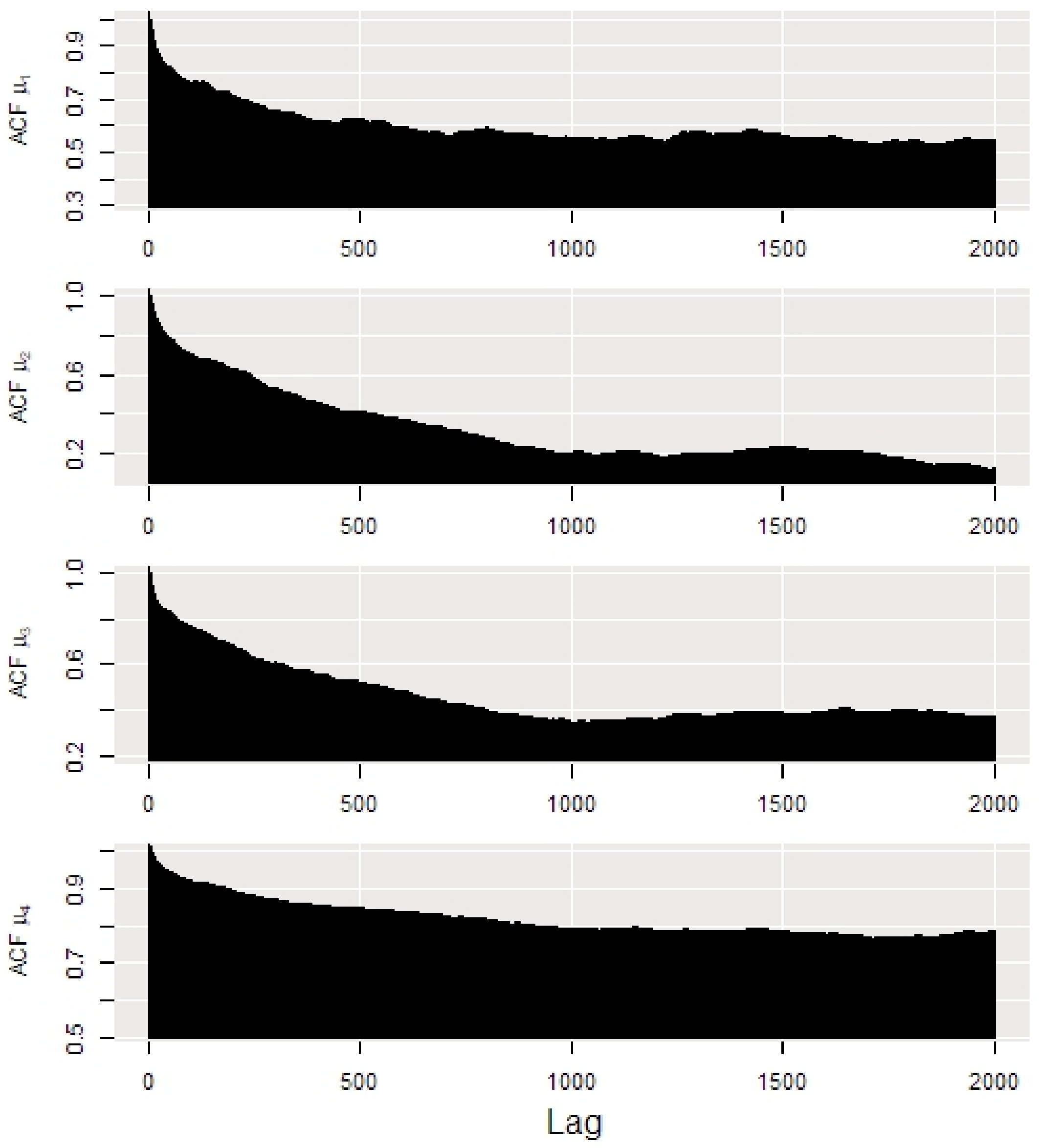

Figure B.12.: Traceplots of the estimates of the means of the four mixture components in LPTC2. The autocorrelation function falls off slowly, since LPTC2 spends more time in different modes than PT2 and LPT2.