Decomposition Sampling applied to Parallelization of Metropolis-Hastings

Jonas Hallgren*† and Timo Koski*

February 13, 2014

Abstract

We consider performing Markov Chain Monte Carlo in parallel. We present an algorithm for sampling random variables which allows us to divide the sampling-process into sub-problems by dividing the sample space into overlapping parts. The sub-problems can be solved independently of each other and are thus well suited for parallelization. Further, on each of these sub-problems we can use distinct and independent sampling methods. In other words, we can design specific samplers for specific parts of the sample space. Moreover we present an algorithm which parallelizes the Metropolis-Hastings algorithm up to the point where it is as fast as it would be with 100%-acceptance rate. The algorithms are demonstrated on a particle marginal Metropolis-Hastings-sampler applied to calibration of a volatility model.

Keywords: Parallel Computing; Metropolis Hastings; MCMC; Stochastic volatility.

1 Introduction

The Metropolis-Hastings (MH) algorithm, by [1953] and [1970], generates a chain which after reaching stationarity produces samples from a specified distribution. A drawback of the otherwise versatile MH-algorithm is that it is unclear how to execute it in parallel since every new iteration depends on the previous one.

A straightforward way to parallelize the algorithm is to simply run many chains in parallel and then, after removing burn-in (the first samples generated, before the chain achieves stationary) combine them in to one single chain. This was done in [2000] and more than ten years later by [2012]. The computing by [2012] is done on

*Department of Mathematics, KTH Royal Institute of Technology, Stockholm SE-100 44, Sweden
†Corresponding author. Email: jonas@math.kth.se
GPU-hardware and is an implementation of the Particle Marginal Metropolis Hastings-sampler (PMMH) which was introduced by Andrieu et al. (2010). This approach can provide an almost linear speed-up (which is ideal) when the convergence rate is fast. In practice however, the rate of convergence can be very slow and it might be precisely that fact which drove the user to parallelization to begin with. Reviewing a few approaches to parallelization reveals that they are often limited to different special cases of MCMC. The Bayesian model learning by Corander et al. (2006) is done on a finite state space. Jacob et al. (2011) presents a method which does parallelize a special case: the independent MH-sampler. Altekar et al. (2004) implements a parallelization of Metropolis coupled MCMC, (MC)$^3$. This approach deals with the fact that (MC)$^3$ is slower than standard MCMC and while the method yields a nice speed-up it is, like Jacob et al. (2011) and Corander et al. (2006), application-specific. The population Monte Carlo method presented by Cappé et al. (2004) and implemented on GPU hardware by Lee et al. (2010) does not suffer from these drawbacks, has many advantages, and is not limited to MCMC.

Other non-MCMC methods for sampling random variables such as importance sampling, rejection-sampling, or inverse transform sampling, all of which are presented by Robert and Casella (2004), are easy to parallelize simply because they are parallel in their nature: no sample is drawn in relation to another.

The algorithm we present is based on a simple idea: decompose the sample space into several parts, subsets, then sample on these subsets independently of each other. If the probability that we land in one subset is higher than in another, we discard some of the samples in the less likely subset. If the probabilities that we land in the subsets are unknown, we can obtain them by evaluating integrals on the intersections of the subsets.

The paper is divided into six sections. The first is this introduction which presents some background and terminology. In Section 2 we introduce necessary nomenclature and objects, and we present the three main algorithms.

1. The first algorithm merges samples drawn in subsets in to a single sample drawn from the full space.

2. The second algorithm produces expectations evaluated on the full space.

3. The third algorithm provides the first two algorithms with the relations between the subsets if the relation is unknown.
In the next section, Section 3, we present an algorithm which is a reformulation of the Metropolis-Hastings sampler. The specific formulation of the algorithm allows us to execute parts of the sampler in parallel. It turns out that the acceptance-probability has almost no influence on the execution time if we have a sufficiently large number of computers working in parallel. This in contrast to the standard formulation of Metropolis-Hastings where the acceptance-probability has a crucial impact on the execution time. In Section 4 we look at an application, where we calibrate a stochastic volatility model using Particle Marginal Metropolis-Hastings. The efficiency of the method is also examined and the results are presented in Section 5. In Section 6 we discuss some improvements that can be made.

2 Decomposition Sampling

Let \( X \) be a random variable on the probability space \((\mathcal{S}, \mathcal{B}(\mathcal{S}), \mathbb{P})\) where \(\mathcal{B}(\mathcal{S})\) is the Borel sigma-algebra of \(\mathcal{S}\). Typically \(\mathcal{S} \subseteq \mathbb{R}^d\) or \(\mathcal{S} \subseteq \mathbb{Z}^d\). Assume that there exist a specific cover \(C = \{C_1, C_2, \ldots, C_W\}\) of \(\mathcal{S}\), such that \(\Delta_j = C_j \cap C_{j+1}\), \(\Delta_0 = \Delta_W = \emptyset\), and \(C_j \cap C_k = \emptyset\), when \(|j - k| > 1\).

That is, assume that there exists a cover such that every element of the cover shares two distinct and exclusive subsets of itself with the previous and following element in the cover. We call such a cover a linked cover of \(\mathcal{S}\). This cover can be constructed by taking a partition of \(\mathcal{S}\), and rearranging the order of the parts so that it is possible to select the intersections as desired. If a linked cover contains only elements and intersections such that the probability that \(X\) is in them is strictly positive with respect to \(\mathbb{P}\), it is said that \(C\) is a saturated linked cover. Assume that we are interested in sampling a random variable taking values in \(\mathcal{S}\). Then an example of a linked cover when \(\mathcal{S}\) is the positive real line is:

\[
C_1 = [0, 3.55], \quad C_2 = [3.45, 7.55], \quad C_3 = [7.45, \infty], \quad (1)
\]
implying that the intersection $\Delta_1$ is given by $[3.45, 3.55]$. Consider an example: we want to generate samples from a gamma-distribution. We use the linked cover from (1). Sampling from each of the subsets in $C$ yields the results presented in the first three subplots of Figure 1. In the fourth subplot the three samples are combined into one and plotted in the same histogram. Clearly, there is an abundance of samples in $C_2$ that would not be there if this truly was a sample from the gamma-distribution. This is where the intersections come into play. An integral evaluated on the intersection $\Delta_1$ using samples from $C_1$ should of course have the same value as an integral evaluated on $\Delta_1$ using samples from $C_2$. This allows us, by looking at $\frac{P(X \in C_1)}{P(X \in C_2)}$, to find out how many of the abundant samples in the tail we should discard. The samples from Figure 1 are processed through the algorithm throwing out the abundant samples, producing the result in Figure 2.

Figure 2: Processed samples from Figure 1
An example of a discrete space linked cover is seen in Figure 3. We assume a particularly vicious transition probability: it is very difficult to reach state 4 from state 3 and very easy to reach it from state 5. This means that if we want to simulate a Markov-chain and we start in state 1, 2, 3 or 4, it is very unlikely that we reach 5, 6, or 7 in a small number of steps. Indeed, simulations indicate that starting in state 4, we need copious amounts of steps to reach state 5. However, if we split the space in two parts, $C_1$ and $C_2$, then we can run simulations exploring the full space in a small number of steps (and in addition, we can do it in parallel!). More details can be found in Section 5 with the other numerical results.

The integer $W$ will be used throughout the paper to denote the number of agents we use. An agent is a machine which can execute an algorithm. In practice a computer typically produces one agent for every core of the CPU, but an agent could also be a cluster of computers. A single-core processor could simulate several agents.

The basis for the algorithm rests on the identity provided by the following proposition:

**Proposition 2.1.** Let $C = \{C_1, \ldots, C_W\}$ be a saturated linked cover of $S$. Then the probability of an event $A \in \mathcal{B}(S)$ can be written

$$
P(X \in A) = \sum_{j=1}^{W} P(X \in A \setminus \Delta_{j-1} \mid X \in C_j) \omega_j,
$$

(2)

where the backslash-symbol denotes the set difference, and $\omega_j$ is defined as:

$$
\omega_j \triangleq P(X \in C_j).
$$

(3)
Further, when \( j > 1 \) we have the relation:

\[
\omega_j = \mathbb{P}(X \in C_{j-1}) \frac{\mathbb{P}(X \in \Delta_{j-1} \mid X \in C_{j-1})}{\mathbb{P}(X \in \Delta_{j-1} \mid X \in C_j)}. \tag{4}
\]

Proof. Observe that

\[
\mathbb{P}(X \in \Delta_j) = \mathbb{P}(X \in C_j \cap X \in \Delta_j). \tag{5}
\]

From this identity it follows that the recursive expression from (4) can be extracted by conditioning in the right hand side of (5):

\[
\mathbb{P}(X \in \Delta_{j-1}) = \mathbb{P}(X \in \Delta_{j-1} \mid X \in C_j) \mathbb{P}(X \in C_j) = \mathbb{P}(X \in \Delta_{j-1} \mid X \in C_{j-1}) \mathbb{P}(X \in C_{j-1}). \tag{6}
\]

We obtain (4) by dividing,

\[
\mathbb{P}(X \in C_j) = \mathbb{P}(X \in C_{j-1}) \frac{\mathbb{P}(X \in \Delta_{j-1} \mid X \in C_{j-1})}{\mathbb{P}(X \in \Delta_{j-1} \mid X \in C_j)}. \tag{7}
\]

Since \( A = \bigcup_{j=1}^{W} (A \setminus \Delta_{j-1}) \cap C_j \) and all the sets in the union are disjoint we can split the probability in to the sum by rewriting:

\[
\mathbb{P}(X \in A) = \sum_{j=1}^{W} \mathbb{P}(X \in (A \setminus \Delta_{j-1}) \cap C_j) = \sum_{j=1}^{W} \mathbb{P}(X \in A \setminus \Delta_{j-1} \mid X \in C_j) \mathbb{P}(X \in C_j). \tag{8}
\]

2.1 Merging

Let \( C = \{C_1, \ldots, C_W\} \) be a saturated linked cover of \( S \). Assume that we have drawn samples of \( X \) from the different sets in \( C = \{C_1, \ldots, C_W\} \). Below these independent samples are merged in to one single sample distributed as \( X \). The following algorithm will generate the independent samples \( \xi_{k}^{C_j} \) at each iteration \( k \). The samples will be drawn from a density \( q_k^{C_j} \) which may depend on \( \xi_{1:k-1}^{C_j} \). That is, each sample from \( C_1, C_2, \ldots, C_W \) may be generated by different densities. In the current setting they are allowed to depend on their individual history only. Interactions between the samples might be possible but would render batch-running impossible. For a Metropolis-Hastings-style sampler we would typically have \( q_k^{C_j}(\cdot) = r^{C_j}(\cdot \mid \xi_{k-1}^{C_j}), \) where \( r \) is a transition
density. A sequence \( \eta = \{ \eta_k \}_{k \geq 1} \), taking values in the full space \( S \) is generated by merging the \( \xi \)-samples. This is done in Algorithm 1 below:

**Algorithm 1** Decomposition sampling

**Initialize** \( \eta = \emptyset \).

For \( k = 1, \ldots, M \)

**Initialize** \( \tilde{\eta} = \emptyset \)

1. For \( j = 1, \ldots, W \)
   
   (a) Sample: \( \xi_{C_j}^k \sim q_{C_j}^k (\cdot) \).
   
   (b) Downsampling: With probability
   \[
   \frac{\mathbb{P}(X \in C_j)}{\max_r \mathbb{P}(X \in C_r)} \mathbb{I}(\xi_{C_j}^k \not\in \Delta_{j-1})
   \]
   put \( \tilde{\eta} \leftarrow \{ \tilde{\eta}, \xi_{C_j}^k \} \).

2. Randomize the order of the samples \( \tilde{\eta} \). Put them in the main sequence: \( \eta \leftarrow \{ \eta, \tilde{\eta} \} \).

We can see that for a finite \( M \) it is possible to first sample the \( \xi \)-trajectories in batch and then, given these \( \xi \)'s merge them into a single \( \eta \)-trajectory. Usually, one would prefer this batch-style sampling to running the algorithm on-line; although both approaches permit the different \( \xi \)-samples to be drawn in parallel.

To be able to state the second part of the next proposition we need to refer to the concept of irreducibility.

Let \( \{ \xi_k \}_{k \geq 1} \) be a Markov-chain. If for each set \( A \in \mathcal{B}(S) \) with \( \pi(A) > 0 \) there exists \( n \) such that \( \mathbb{P}(\xi_n \in A) > 0 \) for every starting value of the chain, we say that the chain is \( \pi \)-irreducible.

Hereafter, let \( \pi(A) = \mathbb{P}(X \in A) \) and \( \pi(A \mid B) = \mathbb{P}(X \in A \mid X \in B) \).

**Proposition 2.2.** Let \( \eta \) be a sample from Algorithm 1

1. If \( \int q_{C_j}^k (x) \, dx = \pi (\cdot | C_j) \) for every \( j \) and for every \( k \), then \( \eta \) is a sample from \( \mathbb{P} \).

2. Let each \( \xi_{C_j}^k \) be \( \pi (\cdot | C_j) \)-irreducible, let \( \zeta \) be a Metropolis-Hastings chain of length \( M \) from the full space \( S \), and let \( \mu \) be a probability distribution. Then for any \( \zeta \) it is possible to
construct an \( \eta \) such that the total variation norm of the distribution of the \( M \)'th draw of \( \eta \),
\[
\pi^M_\eta (\cdot) = \int \mathbb{P}(\eta_M \in \cdot \mid x) \mu(dx),
\]
with respect to \( \pi \) satisfies the following inequality for \( \pi_\zeta \):
\[
\| \pi^M_\eta - \pi \|_{TV} \leq \| \pi^M_\zeta - \pi \|_{TV} \sum_{s=1}^{W} \mathbb{P}(X \in C_s),
\]
(10)
this expression tends to zero as \( M \) tends to infinity.

Further, let \( \omega_j^{(M)} \) be an estimate such that \( \| \omega_j^{(M)} - \mathbb{P}(X \in C_j) \|_{TV} \) tends to zero as \( M \) tends to infinity for every \( j \). Then the distribution \( \pi^M_\eta \) of the \( M \)'th draw of \( \eta \) converges in total variation to that of \( \pi \), that is:
\[
\| \pi^M_\eta - \pi \|_{TV} \xrightarrow[M \to \infty]{} 0.
\]
(11)

By Robert and Casella (2004) a chain generated from the Metropolis-Hastings algorithm is \( \pi \)-irreducible under some mild conditions on the transition kernel. Thus, if each \( \xi^{C_j} \) is a Metropolis-Hastings-chain, then (10) and (11) hold. The next algorithm is used to evaluate expectations of the type \( \mathbb{E} h(X) \) where \( h \) is a function such that the expectation is finite.

**Algorithm 2** Evaluating Expectations

For \( j = 1, \ldots, W \)

1. Sample a sequence of i.i.d. random variables: \( \{\xi^{C_j}_{k}\}_{k=1}^{M} \sim \pi(\cdot \mid C_j) \)
2. Put \( \eta_k \leftarrow \sum_{j=1}^{W} h(\xi^{C_j}_{k}) \mathbb{I}(\{\xi^{C_j}_{k} \notin \Delta_{j-1}\}) \omega_j \).
Algorithm 3 Estimating $\omega$

1. For $j = 1, \ldots, W$
   
   \begin{enumerate}
   \item[i)] Sample an i.i.d. sequence $\{\xi^{C_j}_k\}_{k=1}^M \sim \pi(\cdot | C_j)$
   \item[ii)] If $j = 1$ put $\omega_j^{(M)} \propto 1$. If $j > 1$ put
   \[
   \omega_j^{(M)} \leftarrow \omega_{j-1}^{(M)} \frac{\sum_{k=1}^M \mathbb{I}(\xi^{C_j}_{k-1} \in \Delta_{j-1})}{\sum_{k=1}^M \mathbb{I}(\xi^{C_j}_k \in \Delta_{j-1})}.
   \] (12)
   \end{enumerate}

2. Normalize:
   \[
   \hat{\omega}_j^{(M)} \leftarrow \frac{\omega_j^{(M)}}{\sum_{s=1}^W \omega_s^{(M)} \{1 - M^{-1} \sum_{k=1}^M \mathbb{I}(\xi^{C_j}_k \in \Delta_{j-1})\}}.
   \] (13)

Proposition 2.3. Let $\hat{\omega}^{(M)} = \{\hat{\omega}_1^{(M)}, \ldots, \hat{\omega}_W^{(M)}\}$ be a sample from Algorithm 3. Then, almost surely, for each $j$,

1. \[
   \hat{\omega}_j^{(M)} \xrightarrow{M\to\infty} \mathbb{P}(X \in C_j),
   \] (14)

2. \[
   \tilde{\omega}_j^{(M)} = \hat{\omega}_j^{(M)} \frac{1}{M} \sum_{k=1}^M \mathbb{I}(\xi^{C_j}_k \notin \Delta_{j-1}) \xrightarrow{M\to\infty} \mathbb{P}(X \in C_j \setminus \Delta_{j-1}).
   \] (15)

Further if for every $j$, $\xi^{C_j}$ is a MH-chain which is $\pi(\cdot | C_j)$-irreducible, then (14) and (15) still hold.

This implies that we can get all the required probabilities by looking at the overlaps. A slightly more general version can be obtained. If each subsample $\xi^{C_j}$ has a unique $M^{C_j}$, then the same result is obtained if we use the mean value in the update step (12) of $\omega_j^{(M)}$ above by letting the smallest such $M^{C_j}$ tend to infinity. That is, the update becomes

\[
\omega_j^{(M)} = \omega_{j-1}^{(M)} \frac{(M^{C_j-1})^{-1} \sum_{k=1}^{M^{C_j-1}} \mathbb{I}(\xi^{C_j-1}_k \in \Delta_{j-1})}{(M^{C_j})^{-1} \sum_{k=1}^{M^{C_j}} \mathbb{I}(\xi^{C_j}_k \in \Delta_j)}.
\] (16)
Proposition 2.4. If $\omega^{(M)}$ is estimated from Algorithm 3 and $\eta$ is a sample from Algorithm 2 of size $M$, using the estimate $\omega^{(M)}$, then as $M$ tend to infinity,

$$
\frac{1}{M} \sum_{k=1}^{M} \eta_k \rightarrow \mathbb{E} h(X),
$$

(17)

almost surely. Further if for every $j$, $h \in L^1[\pi(\cdot \mid C_j)]$ and $\xi^{C_j}$ is a MH-chain which is $\pi(\cdot \mid C_j)$-irreducible, then (17) still holds.

When we have the true probabilities $\omega$, and we do not need to estimate them, the proposition still holds.

In this paper we present the MCMC-results in the form of convergence in total variation, and almost sure convergence. For a concise overview of these concepts and alternatives such as a Central Limit Theorem see Nummelin (2002).

3 An Alternative Algorithm

We briefly present an algorithm for parallelizing the Metropolis-Hastings sampler. The algorithm allows us to approach the same speed as the Gibbs sampler which accepts every sample. In other words, we do not have to wait for an accepted proposal. We do this in the setting of the standard Metropolis-Hastings sampler. In the literature the algorithm is formulated as “first sample a variable, then check if it is accepted”. Our formulation may seem odd in the sense that instead of doing this two-step procedure, we sample a binary variable which tells us if we accepted the sample or not. This binary variable is denoted by $I_*$. The formulation is presented in Algorithm 4 below and is equivalent to the Metropolis-Hastings algorithm. With standard terminology, $p(x)$ is our target distribution, which we wish to acquire samples from. The proposal distribution is one from which we can acquire samples. It takes the parameter $x$ and its density function is denoted by $r(\cdot \mid x)$. 
Algorithm 4 Metropolis-Hastings

1. Choose $\xi_0$ in arbitrary way.

2. For $k = 1, \ldots, M$

   (a) Sample $(\xi^*, I^*) \sim r(\cdot \mid \xi_{k-1})$, where $I^*$ is 1 with probability

   $\frac{1 \wedge r(\xi_{k-1} \mid \xi^*) p(\xi^*)}{r(\xi^* \mid \xi_{k-1}) p(\xi_{k-1})}$,

   and zero otherwise.

   (b) If $I^*$ is equal to 1 put $\xi_k \leftarrow \xi^*$ otherwise put $\xi_k \leftarrow \xi_{k-1}$.

Writing it this way, it becomes apparent that it is possible to parallelize at least part of the procedure. The part than can be parallelized is the sampling part, it does not depend on anything else than the previous step and does not change until one sample is accepted. To illustrate this assume that we have $W$ number of agents. If we decide a specific indexing of these agents we get some sort of “time”-aspect where we can pretend that we sampled from one before the other, see Figure 3. The procedure is now similar and presented in Algorithm 5.

While the conclusion that the two algorithms produce equivalent results, might be obvious, we will include the proposition and proof.

Proposition 3.1. Drawing a sample from the parallel Metropolis-Hastings sampler in Algorithm 5 is equivalent to drawing a sample from the standard Metropolis-Hastings sampler from Algorithm 4.

Proof. The standard MH draws samples which are independent and identically distributed until one sample is accepted. The samples in step (19) of the parallel MH are all drawn independently and are thus samples of the type found in the standard sampler. When a sample is accepted in the parallel sampler, all the remaining ones are thrown away as if they never existed. In other words, the two procedures are identical: draw independent samples until we accept a sample. Then, when a sample is accepted, start over doing the same thing again.

An example is considered in Figure 3. Here the number of agents $W$ is set to 3. We run four iterations of the algorithm besides the initialization. The samples that are outputs of the algorithm
Algorithm 5 Parallel MH-sampling

1. Choose $\xi_0$ in arbitrary way and set $k \leftarrow 1$.

2. While $k \leq M$

   (a) Sample:

   $$(\xi^j_*, \mathbb{I}^j_*)^{1:W} \sim r(\cdot \mid \xi_{k-1}),$$

   where $\mathbb{I}^j_*$ is $1$ with probability

   $$1 \wedge \frac{r(\xi_{k-1} \mid \xi^j_*) p(\xi^j_*)}{r(\xi^j_*) p(\xi_{k-1})},$$

   and zero otherwise.

   (b) If $\mathbb{I}^j_*$ is zero for every $j$ then $\xi_{k:k-1+W} \leftarrow \xi_{k-1}$.

   Otherwise $J \leftarrow \arg \min_j \mathbb{I}^j_*$ and then $\xi_{k:k+J} \leftarrow \xi_{k-1}^{1:J-1}$ and $\xi_{k-1+J} \leftarrow \xi^*_J$.

   (c) Update $k \leftarrow k + J$.

are shaded. We can follow the chain of outputs through the solid lines, i.e. the algorithm produced 9 samples plus one from the initialization. In the figure, a diamond indicates that a proposal was accepted, in other words it was the first sample in the iteration to have an indicator variable equal to one. All the samples are drawn from $r(\cdot \mid \xi)$ for some $\xi$. The dashed lines show which $\xi$ was used in the conditioning. For example, every proposal sample in the second iteration is drawn conditioned on the second proposal in the first iteration which in turn is drawn from $r(\cdot \mid \xi_0)$. We consider all possible cases: in the first iteration the second

\[\text{Figure 4: Example run of Algorithm } 5\]
sample is accepted and the third is thus discarded, in the second iteration the first sample is accepted and the two others are discarded. In the third iteration, no sample is accepted which means that we can utilize all our proposed samples, this is also the case in the last iteration where we accept the last sample. When \( W \) is larger than 3, the first situation would be different in the sense that the accepted sample could be anywhere between the first and last, the other cases would remain unchanged.

### 3.1 Performance

For a fixed acceptance rate, the probability that we will accept a sample follows a geometric distribution. If we let \( \gamma \) denote the number of samples we have to draw until a sample is accepted then we have \( \Pr(\gamma \leq W) = \sum_{k=1}^{W} a(1 - a)^{k-1} = 1 - (1 - a)^{W} \). Thus as the number of agents \( W \) grows large the probability that we will accept a sample from one of the first \( W \) candidates becomes large. This means that instead of expecting to do \( 1/a \) iterations in sequence before we can accept a sample, we expect to do them all in single parallel iteration. We assume that when one is interested in these types of schemes then the sampling-part is computationally expensive. This is very much the case for the Particle Marginal Metropolis-Hastings sampler implemented in the next section. This assumption suggests that the overhead costs (that is, the extra cost for implementing the parallelization compared to running the standard algorithm) are considered to be negligible.

## 4 Applications

### 4.1 Parallelization of Particle Marginal MH-sampler

The Particle Marginal Metropolis-Hastings (PMMH)-sampler was introduced by Andrieu et al. (2010). The algorithm can for instance solve problems in parameter-estimation that were previously unsolved, and it is fairly easy to implement. However, as with other MCMC-methods it is not trivial to parallelize it. The Decomposition sampling can parallelize any random-sampling approach and we choose to demonstrate the algorithm on the PMMH-sampler. We will apply it to a problem of calibrating a stochastic volatility model. We model the variation (or the volatility) of the price of a financial instrument. We do this in discrete time where we observe the instrument
price $S_k$ at time $k$. We will model the log-returns, defined as: $Y_k = \log(\frac{S_k}{S_{k-1}})$. Our model for these log-returns is as follows:

$$
Y_k = \beta e^{X_k u_k},
X_k = \phi X_{k-1} + \sigma w_k,
$$

(21)

where $u_k$ and $w_k$ are jointly normally distributed with zero mean, unit variance and correlation $\rho$. Our object is now to estimate the four parameters $\phi, \beta, \rho$ and $\sigma$. We will do this with a Bayesian approach. Since we can not observe $X$ we will use sequential Monte Carlo methods to obtain estimates of it. More specifically, we will use the Particle Marginal Metropolis-Hastings (PMMH) sampler, see Andrieu et al. (2010) or Olsson and Rydén (2011) for an extension. The example is taken from Hallgren (2011) where the posteriors and choice of priors can be found. The model in (21) is a variation of one introduced in Taylor (1982), where $u$ and $w$ has zero correlation, i.e. $\rho$ is equal to zero. For more on the history of the model and an economic interpretation, see Shephard (2005). In Hallgren (2011) the model is evaluated on market-data, the results indicate that it performs better (with respect to value at risk and expected shortfall) than the model without correlation. In this paper we will however restrict ourselves to fitting the parameters. We use the Bayesian approach and the variable we are interested in sampling is $Z = (\phi, \beta, \rho, \sigma, X_{0:n})$. In the setting of the PMMH-sampler $X$ is needed to to get estimates of the other parameters. It
is natural to let $\sigma$ be greater than zero and to let the modulus of $\rho$ be smaller than 1. We note that switching the sign of $\beta$ is the same thing as switching the sign of $u$, which in turn implies that we switch the sign of $\rho$. Thus it is natural to keep $\beta$ positive (or negative). Further we view $X$ as some sort of auto-regressive volatility. This combined with empirical studies, found in Hallgren (2011), engenders a positive $\phi$; it is forced to be smaller than 1 to maintain stationarity. Summarized:

$$S^Z = \{[0, 1), \mathbb{R}^+, [-1, 1], \mathbb{R}^+, \mathbb{R}^n\}. \quad (22)$$

First we observe that, under some assumptions, Theorem 4 in Andrieu et al. (2010) implies that we are dealing with a Metropolis-Hastings chain which we can parallelize in our setting. In order to apply the Decomposition sampling, Algorithm 11 we need to divide the space $S^Z$ into subsets. We choose a simple split, i.e. we divide the space into two parts, $C_1$ and $C_2$. We will choose the parts in a heuristic way; in Section 6 we will discuss more sophisticated methods for choosing the splits, although none of them will be implemented or investigated in this paper. We divide the
space in the $\phi$-variable in the following way:

$$C_1 = \{[0, 0.55 + 0.01], R^+, [-1, 1], R^+, R^n\},$$

$$C_2 = \{[0.55 - 0.01, 1], R^+, [-1, 1], R^+, R^n\},$$

thus the intersection $\Delta$ is given by $\Delta = \{[0.54, 0.56], R^+, [-1, 1], R^+, R^n\}$. It is now possible to apply the Decomposition sampling combined with the Proportionality algorithm. We fit the model to log-returns of the price of the Disney (NYSE: DIS) stock. In figures 5 and 6 we see the resulting samples from the two subsets in (23). By processing the data through the two algorithms we obtain the final sample in Figure 7.

![Figure 7: Processed samples from Figure 5](image)

5 Results

In this section we will present some execution-times and comparisons, we start with the discrete-spaced example from the introduction. This illustrates the advantages of decomposing a state space when it is difficult to move from one part to another. The second case is the Particle Marginal Metropolis-Hastings sampler applied to calibration of the volatility model from the
previous section. It utilizes the properties of the algorithm to gain more samples in less time by running two chains in parallel.

5.1 Discrete Model

Consider a Markov-chain moving on the discrete space according to Figure 3. We wish is to obtain samples from this distribution. We can implement this as a MH-sampler. We use the kernel of the chain as a proposal-kernel. Since the chain is reversible, see Häggström (2002), we will have acceptance rate 1 for the MH-sampler. In other words, the MH-sampler collapses to simply simulating the Markov-chain. In the case when we split the chain, the rejection of a proposal always leads back to state 4.

Running the simulations, we find that the standard MH-sampler, even if started with the stationary distribution, usually does not find its way to the higher numbered states. In Table 1 we compare the empirical total variation between the actual stationary distribution $\lambda$ and our estimate $\hat{\lambda}$:

$$
\|\lambda - \hat{\lambda}\|_{TV} = \max_j \left| \frac{1}{N} \sum_{k=1}^{N} \mathbb{I}(\xi_k = j) - \lambda_j \right|,
$$

(24)

where $\{\xi_k\}_{k=1}^{N}$ is the generated chain. In the column “states”, we register which states the chain visited. Clearly, the Decomposition sampling, denoted by MH-DC in the table, is superior to the standard MH-method. We can obtain a better result in less than a second with the parallel version than we can in an hour with the standard-version. The results above the line comes from the vicious distribution mentioned in the introduction while the results below the line originates from a slightly more pleasant distribution. There are 10% more samples from the parallel chains, these 10% are the extra samples gained from the split. It is clear that the gain in terms of number of samples is quite small, it could have been made larger by splitting in another state than 4; however the purpose of this experiment was to demonstrate other ad-

![Figure 8: Hastings example](image-url)
Table 1: Simulation results for the discrete space model

| Method   | N       | Runtime (s) | \(\|\lambda - \hat{\lambda}\|_{TV}\) | States |
|----------|---------|-------------|--------------------------------------|--------|
| MH-Standard | 10^8    | 9898        | 0.0008                               | All    |
| MH-Standard | 10^7    | 879         | 0.1001                               | 1,2,3,4|
| MH-DC    | 1.1 \times 10^4 | 0.918   | 0.0090                               | All    |
| MH-Standard | 10^7    | 961         | 0.0108                               | All    |
| MH-Standard | 10^6    | 90.8        | 0.0736                               | All    |
| MH-DC    | 1.1 \times 10^7 | 916    | 0.0002                               | All    |
| MH-DC    | 1.1 \times 10^3 | 0.094   | 0.0107                               | All    |

vantages of the algorithm than the extra samples it provides.

As another demonstration we reproduce the example from the original paper by Hastings (1970) where a MH-sampler is implemented to generate samples from a Poisson-distribution. We run 1000 iterations, with the intensity set to 14. We split the state space in two parts, the mode of the distribution is the point in which we split, and find that the total variation is significantly smaller for the Decomposition-sampler than for the MH-sampler. Repeating 10^5 such iterations yields the estimate of the TV-norm in Figure 8. Allowing the MH-sampler to run for 2000 iterations produces a total variation similar (but still worse) to that of Decomposition-sampler; however this means that the execution time is four times longer.

5.2 Calibration of a Stochastic Volatility Model

We use the split from (23) and compare the result to running a standard MH-sampler. The results are presented in Table 2. Note that the results for MH-Parallel are estimated results. The estimates are done by executing the parallel method as it would have been executed if the input was from the MH-standard run. The estimate is then \(\frac{\text{Iterations in MH}}{\text{Iterations in MH-parallel}}\). An interesting observation was that 50% of the samples were accepted within 3 steps. This means two things: first that we are wasting a lot of computational resources by drawing a thousand samples and throwing 997 away. But more pleasingly, this also implies that the speedup is much greater than (average acceptance rate)^{-1}. This since in most cases the acceptance probability is quite high while in the cases when it is very low, the speedup is very large. In the MH-standard run for
Table 2: Simulation results for the volatility model

| Method          | Data       | N     | W  | Runtime (s) |
|-----------------|------------|-------|----|-------------|
| MH-Standard     | Simulated  | $1.6 \cdot 10^6$ | 1  | 18,139      |
| MH-DC           | Simulated  | $1.6 \cdot 10^6$ | 2  | 10,845      |
| MH-Standard     | NYSE: DIS  | $1.2 \cdot 10^6$ | 1  | 12,240      |
| MH-DC           | NYSE: DIS  | $1.2 \cdot 10^6$ | 2  | 10,635      |
| MH-Parallel     | NYSE: DIS  | $1.2 \cdot 10^6$ | 100| 800         |
| MH-Parallel     | NYSE: DIS  | $1.2 \cdot 10^6$ | 6  | 2,800       |
| MH-Parallel     | NYSE: DIS  | $1.2 \cdot 10^6$ | 2  | 6,600       |

example the average acceptance speedup would be roughly of factor 8, meaning that we have an average acceptance rate of about an eighth; however the actual speedup for a large number of agents was almost of factor 30. In Figure 9 we see the estimated speedup (defined to be number of loops needed for the parallel method divided by the number of loops for the standard method) for the MH-standard run. The function plotted is a constant $k$ multiplied by the logarithm of the number of agents $W$: $k \cdot \log(W) + 1$. The constant was in this case found through the least squares estimate to be roughly 3. The constant is data-dependent. It should be noted that the log-function is a decent approximation for a small number of agents. As the number of agents grow large, we expect the speedup to converge to its upper bound which is limited by the number of accepted samples.

6 Discussion and Conclusions

The algorithm can have at least two interesting uses: parallelization and custom designed samplers. The parallelization yields, if the cover $C$ is carefully selected, more samples. Sometimes, as is very much the case with Hastings example, the gain is not in the number of samples we get but in the polytropic exploration of the state space.

Figure 9: Estimated speedup
An interesting development would be an automation of the selection of $C$. Using rejection sampling on multimodal distributions would be an interesting application. Simulating random processes would be another.

## A Proofs

### A.1 Proof of Proposition 2.2

**i)** We will examine an arbitrarily selected $k$. First we note that in the downsampling step, we multiply the acceptance-probability with the indicator function $\mathbb{I}(\xi^C_j \not\in \Delta_{j-1})$, which is the same thing as forcing our draw to be conditionally from $C_j \setminus \Delta_{j-1}$. In other words: if we evaluate a probability of an event $A$ we do it conditioned that we are not in the intersection $\Delta_{j-1}$.

Thus $\P(\eta_k \in A) = \sum_{j=1}^{W} \P(\xi^C_j \in A \setminus \Delta_{j-1}) \P(X \in C_j)$, where we use that the probability that a distinct $\eta_k$ comes from a certain $C_j$ is precisely $\P(X \in C_j)$. Since all the $\xi^C_j$'s are drawn from the conditional distributions $X \mid X \in C_j$, we have

$$\P(\eta_k \in A) = \sum_{j=1}^{W} \P(X \in A \setminus \Delta_{j-1} \mid X \in C_j) \P(X \in C_j). \quad (25)$$

Since this coincides with the expression in (2) from Proposition 2.1 we end up with $\P(\eta_k \in A) = \P(X \in A)$ for an arbitrarily selected $k$ and the property is established.

**ii)** Recall that $\pi(A) = \P(X \in A)$. We want to show:

(a) $\pi^M_\eta \xrightarrow{TV} \pi$

(b) $\|\pi^M_\eta - \pi\|_{TV} \leq \|\pi^M_\xi - \pi\|_{TV} \sum_{j=1}^{W} \P(X \in C_j)$

Where $\pi^M_\eta(\cdot) = \int \P(\eta_M \in \cdot \mid x)\mu(dx)$, $x$ is a vector of initial values for $\xi$ and $\mu$ is a probability distribution. We prove first (b).

(b) Let $\omega_j$ denote the probability $\P(X \in C_j)$, and let $x_j$ be the initial value of each chain $\xi^C_j$, further let $M_j$ denote the current time of each chain when $\eta$ is at time $M$. Then the
second claim follows from:

\[ \| \pi^M_\eta - \pi \|_{TV} = \sup_A |\pi^M_\eta - \pi| \]

\[ = \sup_A \sum_{j=1}^W \mathbb{P}(\xi^C_j \in A \setminus \Delta_{j-1} \mid x_j)\omega_j - \mathbb{P}(X \in A) \]

\[ = \sup_A \sum_{j=1}^W \left\{ \mathbb{P}(\xi^C_j \in A \setminus \Delta_{j-1} \mid x_j) - \mathbb{P}(X \in A \setminus \Delta_{j-1} \mid X \in C_j) \right\}\omega_j \]

\[ \leq \sup \max_j \left\{ \mathbb{P}(\xi^C_j \in A \setminus \Delta_{j-1} \mid x_j) - \mathbb{P}(X \in A \setminus \Delta_{j-1} \mid X \in C_j) \right\} \sum_{s=1}^W \omega_s \]

\[ = \max_j \| \pi^M_{\xi_j^C \setminus \Delta_{j-1}} - \pi_{C_j \setminus \Delta_{j-1}} \|_{TV} \sum_{s=1}^W \omega_s \leq \| \pi^M_\zeta - \pi \|_{TV} \sum_{s=1}^W \omega_s. \] (26)

The last inequality follows from the fact that \( \zeta \) and \( \xi_{C_j \setminus \Delta_{j-1}} \) can be related. The chain \( \zeta \) is exploring the full space. The chain \( \xi_{C_j \setminus \Delta_{j-1}} \) can be defined to get its values from \( \zeta \) when it is in \( C_j \setminus \Delta_{j-1} \). In that case

\[ \| \pi^M_{\xi_j^C \setminus \Delta_{j-1}} - \pi_{C_j \setminus \Delta_{j-1}} \|_{TV} = \| \pi^M_{\xi_j^C \setminus \Delta_{j-1}} - \pi_{C_j \setminus \Delta_{j-1}} \|_{TV} \leq \| \pi^M_\zeta - \pi \|_{TV} \] (27)

and (b) is established. To establish (a), note that \( M_j \geq \frac{M}{W} \). The equality holds only when we accept every proposal. Since \( W \) is fixed, we realize that \( M_j \) tends to infinity as \( M \) tends to infinity. By Theorem 7.4 in [Robert and Casella (2004)] we know that if \( \zeta \) is a \( \pi \)-irreducible Metropolis-Hastings Markov Chain, then, for every initial distribution of \( x_j \), the norm

\[ \| \pi^M_\zeta - \pi \|_{TV} \] (28)

tends to zero as \( M \) tends to infinity. This implies the claim: (a) the total variation norm \( \| \pi^M_\eta - \pi \|_{TV} \), tends to zero as \( M \) tends to infinity.

Now assume that we do not know \( \omega_j \), and let \( \omega_j^{(M)} \) be an estimate of \( \omega_j \) such that the total variation distance between \( \omega_j^{(M)} \) and \( \omega_j \) tends to zero as \( M \) tends to infinity. Define:

\[ \varepsilon_j^M = \omega_j^{(M)} - \omega_j, \text{ thus } \omega_j^{(M)} = \omega_j + \varepsilon_j^M. \]

We modify the expression from (26) to include

\[ \mathbb{P}(\xi^C_{M,j} \in A \setminus \Delta_{j-1} \mid x_j)\omega_j^{(M)} = \mathbb{P}(\xi^C_{M,j} \in A \setminus \Delta_{j-1} \mid x_j)(\omega_j + \varepsilon_j^M). \] (29)
This gives us:

\[
\|\pi^M_{\eta} - \pi\|_{TV} \leq \sup_A \max_{j} \left\{ \mathbb{P}(\xi^{C_j}_{M_j} \in A \setminus \Delta_{j-1} \mid x_j) \right\} \sum_{s=1}^{W} w_s \\
- \mathbb{P}(X \in A \setminus \Delta_{j-1} \mid X \in C_j) \right\} \\
+ \|\varepsilon^M_{j}\|_{TV} \sum_{s=1}^{W} \mathbb{P}(\xi^{C_s}_{k_s} \in A \setminus \Delta_{s-1} \mid x_s) \\
\leq W \left( \max_j \|\pi^{M_j}_{\xi_c} \setminus \pi_{C_j} \setminus \Delta_{j-1}\|_{TV} + \|\varepsilon^M_{J}\|_{TV} \right) \\
\leq W \left( \|\pi^{M_j}_{\zeta} - \pi\|_{TV} + \max_j \|\varepsilon^M_{j}\|_{TV} \right).
\]

(30)

By (28) \(\|\pi^{M_j}_{\zeta} - \pi\|_{TV}\) tends to zero as \(M\) tends to infinity and by assumption \(\|\varepsilon^M_{j}\|_{TV} = \|\omega^{(M)} - \omega\|_{TV}\) tends to zero as \(M\) tends to infinity. Thus, provided an estimate of \(\omega_j\) which converges to the true value in total variation, the estimate \(\pi^M_{\eta}\) converges in total variation to \(\pi\).

A.2 Proof of Proposition 2.3

This proof relies on the independence of the subsamples.

Proof. Note that the strong LLN, or when \(\xi^{C_j}\) is a \(\pi(\cdot \mid C_j)\)-irreducible MH-chain: Theorem 7.4 in Robert and Casella (2004), implies that

\[
(M)^{-1} \sum_{k=1}^{M} \mathbb{I}(\xi^{C_j}_{k} \in \Delta_{j}) \xrightarrow{M \to \infty} \mathbb{P}(X \in \Delta_{j} \mid X \in C_j),
\]

(31)

almost surely, and thus, by continuity

\[
\frac{\sum_{k=1}^{M} \mathbb{I}(\xi^{C_{j-1}}_{k} \in \Delta_{j})}{\sum_{k=1}^{M} \mathbb{I}(\xi^{C_j}_{k} \in \Delta_{j})} \xrightarrow{M \to \infty} \frac{\mathbb{P}(X \in \Delta_{j} \mid X \in C_{j-1})}{\mathbb{P}(X \in \Delta_{j} \mid X \in C_j)},
\]

(32)

almost surely. Now, put \(\hat{\omega}_1^{(M)} = 1\) and update the other \(\hat{\omega}^{(M)}\)'s. According to equation (4) in Proposition (2) our estimated distribution \(\hat{\omega}^{(M)}\) is now proportional to the true distribution \(\omega\), which is such that \(\omega_j = \mathbb{P}(X \in C_j)\). Since the sets in \(C\) are pairwise intersected, these intersections are counted twice, thus

\[
\sum_{j=1}^{W} \omega_j = 1 + \sum_{j=1}^{W} \mathbb{P}(X \in \Delta_{j-1}) = 1 + \sum_{j=1}^{W} \mathbb{P}(X \in \Delta_{j-1} \mid X \in C_j)\omega_j.
\]

(33)
In other words, the sum \( \sum_{j=1}^{\infty} \omega_j \) can be slightly modified to match a probability distribution, meaning that it sums to 1: \( \sum_{j=1}^{\infty} \omega_j \{1 - \mathbb{P}(X \in \Delta_{j-1} \mid X \in C_j)\} = 1 \), we exploit this relationship to obtain the normalizing constant. This proves the first statement and the convergence in (14).

Denote the complement of a set \( A \) by \( A^* \). Then the second statement in (15) follows from taking the limit of (15) and see that it coincides with:

\[
P(X \in C_j \setminus \Delta_{j-1}) = P(X \in C_j \cap X \in \Delta_{j-1}^*) = P(X \in \Delta_{j-1}^* \mid X \in C_j)P(X \in C_j) = (1 - P(X \in \Delta_{j-1} \mid X \in C_j))P(X \in C_j). \tag{34}
\]

\[\square\]

### A.3 Proof of Proposition 2.4

First observe that

\[
\mathbb{E} h(X) = \sum_{j=1}^{W} \mathbb{E}[h(X) \mid X \in C_j \setminus \Delta_{j-1}] \mathbb{P}(X \in C_j \setminus \Delta_{j-1}) . \tag{35}
\]

Since \( \tilde{\omega}^{(M)} \) is sampled from Algorithm 3 we have almost sure convergence to: \( \mathbb{P}(X \in C_j \setminus \Delta_{j-1}) \).

Further, since \( \xi_k^{C_j} \overset{d}{=} X \mid X \in C_j \) we have by the strong LLN, or in the case when \( \xi_j^{C_j} \) is a \( \pi(\cdot \mid C_j) \)-irreducible MH-chain by Theorem 7.4 in Robert and Casella (2004), that the sum \( \frac{1}{M} \sum_{k=1}^{M} h(\xi_k^{C_j}) \) converges almost surely to \( \mathbb{E}[h(X) \mid X \in C_j] \) as \( M \) tends to infinity. By continuity we have that the product of these two estimates almost surely converges to the product of their limits:

\[
\frac{1}{M} \sum_{k=1}^{M} h(\xi_k^{C_j}) \tilde{\omega}_j^{(M)} \xrightarrow{M \to \infty} \mathbb{E}[h(X) \mid X \in C_j] \mathbb{P}(X \in C_j \setminus \Delta_j) . \tag{36}
\]

Combining (36) with the fact that we for each \( C_j \) discard the samples that are in \( \Delta_{j-1} \), i.e. the \( \xi \)'s we allow to contribute to \( \eta \) are distributed as \( X \mid X \in C_j \setminus \Delta_{j-1} \), yields almost surely for \( \eta \):

\[
\frac{1}{M} \sum_{k=1}^{M} \eta_k \xrightarrow{M \to \infty} \sum_{j=1}^{W} \mathbb{E}[h(X) \mid X \in C_j \setminus \Delta_j] \mathbb{P}(X \in C_j \setminus \Delta_j) . \tag{37}
\]

This coincides with (35) which is what we wanted to establish.
References

Gautam Altekar, Sandhya Dwarkadas, John P. Huelsenbeck, and Fredrik Ronquist. Parallel metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics*, 20(3):407–415, 2004.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3): 269–342, 2010.

Olivier Cappé, Arnaud Guillin, Jean-Michel Marin, and Christian P. Robert. Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13(4), 2004.

Jukka Corander, Mats Gyllenberg, and Timo Koski. Bayesian model learning based on a parallel MCMC strategy. *Statistics and computing*, 16(4):355–362, 2006.

Olle Häggström. *Finite Markov chains and algorithmic applications*, volume 52. Cambridge University Press, 2002.

Jonas Hallgren. *Calibration of Stochastic Volatility models using Particle Markov chain Monte Carlo methods*. Number (2011:E44) in Master’s thesis. Lund University, 2011.

W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Soren Henriksen, Adrian Wills, Thomas B Schön, and Brett Ninness. Parallel implementation of Particle MCMC Methods on a GPU. In *System Identification*, volume 16, pages 1143–1148, 2012.

Pierre Jacob, Christian P. Robert, and Murray H. Smith. Using parallel computation to improve independent Metropolis–Hastings based estimation. *Journal of Computational and Graphical Statistics*, 20(3):616–635, 2011.

Anthony Lee, Christopher Yau, Michael B Giles, Arnaud Doucet, and Christopher C Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 19(4):769–789, 2010.
Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087, 1953.

Esa Nummelin. MC’s for MCMC’ists. *International Statistical Review*, 70(2):215–240, 2002.

Jimmy Olsson and Tobias Rydén. Rao-Blackwellization of particle Markov chain Monte Carlo methods using forward filtering backward sampling. *Signal Processing, IEEE Transactions on*, 59(10):4606–4619, 2011.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

Jeffrey S Rosenthal. Parallel computing and Monte Carlo algorithms. *Far east Journal of Theoretical Statistics*, 4(2):207–236, 2000.

Neil Shephard. Stochastic Volatility. *Economics Group, Nuffield College, University of Oxford, Economics Papers*, 2005.

S.J. Taylor. Financial returns modelled by the product of two stochastic processes – a study of daily sugar prices 1961-79. in o.d. anderson (ed.). *Time Series Analysis: Theory and Practice*, (1):203–226, 1982.