LITERATURE REVIEW:
Parallel Metropolis-Hastings Algorithms

Boyan Bejanov
bbejanov@bankofcanada.ca

February 17, 2014

1 Introduction

The Monte Carlo methods (MC) are a class of numerical integration techniques, where the integral is interpreted as the expected value of some function of some random variable. The expectation can be statistically estimated from a sample drawn from the same distribution as the underlying random variable and this estimate serves as the numerical approximation of the integral. Needless to say, the MC methods are also applicable to problems other than numerical integration, so long as they can be reduced to the evaluation of an expected value.

The successful application of MC relies on the availability of numerical techniques for simulation of random samples from a distribution, which is given by its probability density function (pdf). We will denote this *target* pdf by $\pi(x)$. It is always assumed that a stream of independent random numbers with uniform distribution in the interval $(0, 1)$ is available without ado. For some of the other common probability distributions there are direct methods of simulation, which perform some computation on the uniform random numbers, transforming them into random numbers of the desired distribution. However, these techniques are limited to special cases. One of the most important general sampling technique is the Acceptance-Rejection sampling (AR). Each iteration of the AR goes in two steps. We first simulate a random number, X, from some other distribution, which can be simulated by an already known method. We call this distribution *proposal* and denote its pdf by $q(x)$. In the second step, we simulate a uniform random number and compare it to the adjusted ratio of the target and proposal densities. Based on this comparison, we either accept X as a random number from π, or we reject and discard it. The AR method is embarrassingly parallel, since multiple random numbers can be simulated on multiple processors simultaneously. In addition, it produces a sample of independent random numbers, which can be used directly in an MC method. However, its applicability is limited to the cases where $\pi(x)$ is given explicitly by a known formula. Even then, the computational efficiency of AR (as measured by the acceptance rate) depends on the choice of the proposal, $q(x)$, and a tight estimate of the adjustment constant in the AR test, both of which can require some non-trivial mathematical derivations, especially for multivariate distributions.

The method of choice in the cases of a multivariate distribution of high dimension, or when the formula for the target pdf is too complicated, or when the target pdf can only be computed numerically, is the Markov Chain Monte Carlo method (MCMC). A Markov Chain is a process, which is described by a *state space*, i.e. the set of all possible values the process can take, and a transition rule, which may depend only on the current state, but not any earlier ones. It is known from the theory of Markov chains that, if certain conditions are satisfied, the process has an *invariant distribution* and the distributions of successive states converge to the invariant distribution. In the MCMC method, we construct a transition rule, such that the resulting Markov chain has an invariant distribution, and this invariant distribution is exactly $\pi(x)$. The initial part of the chain, before the distribution of states gets close enough to the invariant distribution, is known as *burn-in* and is discarded. The remaining chain can be used as the random sample in a Monte Carlo method, although with the caveat that it is not an independent random sample, so the auto-correlation of the process must be taken into account.

The AR, MCMC and other methods for simulation of random variables, as well as the underlying mathematical theories, can be found in [12].

The Metropolis-Hastings algorithm (M-H) combines AR and MCMC. Effectively, it is a recipe for constructing the transition rule of a Markov chain given its desired invariant distribution. Once again, we have a proposal distribution, however this time the proposal distribution depends on the current state of the chain, i.e. $q = q(x; y)$. The good news is that the accept-reject test is a ratio of the values of q and π at the current and the proposed points with no adjustment constants to be derived in advance. If the proposed point is accepted, then it becomes the next state of the chain, otherwise the next state is equal to the current state, i.e. no points are discarded as it was the case in the AR sampling method. An intuitive and self-contained presentation of the Metropolis-Hastings method can be found in [4].

Other than Monte Carlo type estimations of expected values, the M-H is a general method for simulating random samples from distributions that are otherwise impossible to simulate. It is particularly popular for Bayesian analysis of complex stochastic models. Such models always involve a number of parameters, which rarely can be derived from first principles. More frequently, the parameters are estimated from observed data using maximum-likelihood, or some other methods of statistical estimation. When the model is sufficiently complex, the usual inference based on confidence intervals for the estimated parameters is impossible to derive. In such cases, the approach of Bayesian statistics is to consider the parameters as random variables, simulate a sample from their joint distribution, and use this sample to infer the properties of the distribution that are of interest.

The specific focus in this project is the application of M-H to Bayesian inference of large Dynamic Stochastic General Equilibrium models (DSGE), which are used in the field of macro-economics ([15]). The likelihood function is computed using a Kalman filter, which can be computationally very expensive, especially in the nonlinear case. Therefore, when considering various methods, we will keep in mind the (simplifying) assumption that the evaluation of $\pi(x)$ is much more time consuming than all the other computations in a single step of the Markov chain (such as the computation of $q(x, y)$, simulation from $q(x, y)$, individual arithmetic operations).

The objectives of this project include

- Survey the existing methods and evaluate them.
- Identify the most promising method, for the particular setting we have, and implement it.
- Investigate the benefits of parallelisation in terms of theoretical and practically achieved speedup.
- Consider the question of P-completeness of the M-H algorithm.

2 Literature Review

The search for parallel algorithms for M-H and other MCMC techniques has received much attention from the computational statistics community as well as from researchers in various fields where MCMC finds real-world applications. Unfortunately, there seems to be little attention given to this question in the world of computer science.

In [13] the authors propose an *adaptive* M-H method for estimation of a climate model, where multiple Markov chains are simulated in parallel. In an adaptive MCMC algorithm, the computed values of the target pdf are used to construct a better proposal distribution. Each chain must undergo its own burn-in, therefore the time before the algorithm starts yielding useful output is the same as in the sequential case. This greatly limits the gain in speedup due to parallelisation. However, the periodic updates to the proposal distribution in each chain use the values of the target density produced by all chains. This way the algorithm produces higher quality proposal compared to the sequential version, thus improving the *statistical efficiency* of the chain.

Another recent application of parallel MCMC is presented in [17] in the context of Bayesian models of animal breeding and genetics. Here the authors also consider multiple Markov chains running in parallel and acknowledge that this is only applicable to single-parameter models and simpler models with few parameters, where the burn-in is small. In the case of more complex models, they use parallel computation of $\pi(x)$ by leveraging specific properties of the particular model they consider.

A modified M-H algorithm is presented in [9]. It simulates a single Markov chain with improved statistical properties by considering multiple proposal points at each step. The proposals are computed in parallel and one of them is randomly selected to be the next state. This allows for better coverage of the state space, i.e. reduced burn-in time. However, the time for computing one step is the same as in the sequential M-H algorithm.

In [16], the authors propose a parallel MCMC method which is only applicable to numerical integration. The state space is partitioned into disjoint regions and the total integral is evaluated as the sum of the integrals over the individual regions. A separate Markov chain is simulated for each region and all chains run in parallel. The clever innovation of this method is that it allows the chains to leave their respective regions and are recombined afterwards. This considerably simplifies previous methods using partitioning, which must contain the individual chains within their regions. Unfortunately, this method is not applicable in our case, since it only computes the expectation, but doesn't produce a random sample.

In general the parallel MCMC methods can be divided into methods using multiple independent chains that are simulated in parallel and methods that simulate a single chain in parallel. All of the methods discussed so far are in the first category. The paper [7] contains a recent and detailed overview of parallel MCMC methods based on multiple chains.

The single chain parallel methods can be further subdivided into *within draw* and *between draw* parallelisation, as suggested in [14]. In the former class of methods multiple processors collaborate on a single computation of the target density. Clearly this approach is problem-specific. The between draw parallel methods are characterized by considering multiple future steps of the chain and pre-computing the necessary values of the target density in parallel.

A Metropolis-Hastings algorithm whose proposal density does not depend on the current state of the Markov chain is called *Independence* Metropolis-Hastings (IMH). In this case the between-draw parallelisation is straight-forward. Unfortunately, a good independence proposal is difficult to construct beforehand. Instead, in the method proposed in [15], an independence proposal is estimated from the values of π that have already been computed. As the chain runs, the independence proposal is adapted to any new information about the target density. However, the Markov chain must start with a non-independence proposal. At each adaptation point, the adapted proposal is constructed as a mixture of the newly re-estimated independence proposal and the non-independence proposal, with the relative weight of the independence proposal gradually increasing. See also [8] and [6], which suggest similar algorithms with different approaches to constructing the independence proposal. Many other examples of adaptive MCMC can be found in [11].

A regeneration time in a Markov chain is a time when the state is independent of the history of the chain up to that point. The pieces of the chain between two regeneration times are called *tours* and are independent and identically distributed entities. This allows multiple tours of the chain to be simulated in parallel, each starting from a regeneration time and running until the next recreation time is encountered. The tours are then concatenated to form a single long chain. Although a wide class of Markov chain are regenerative, the identification of regeneration points is very difficult (cf. [10] and [5]). More recently, [2] proposed a constructive method of identifying regeneration times in the context of parallel MH. Using regeneration is only advantageous for low- and moderately-dimensional distribution, because the expected length of the tour increases dramatically with dimension.

A recent algorithm, proposed in [1], is called pre-fetching. Suppose the Markov chain of an MH algorithm is at some state X_n at some time n. All possible trajectories for the next h steps form a full binary tree with root X_n and depth h. At each node the two edges correspond to ”accept” and ”reject”. There are 2^h possible paths and a total of 2^h different proposal states in the tree. The idea of pre-fetching is to compute the values of π at the 2^h proposals in parallel and then perform the h accept-reject steps sequentially on the master process. In [14] the authors suggest an improvement of this algorithm, whereby only the most probable paths are pre-fetched, thus reducing the number of required processors, although at the risk that fewer than h may be completed. Another method suggested in [3] can be interpreted as a special case of this methodology suggested in [14]. The authors leverage the fact that the optimal acceptance rate (from probabilistic point of view) is usually low and pre-fetch only the ”reject” branches.

References

[1] AE Brockwell. Parallel Markov Chain Monte Carlo simulation by pre-fetching. *Journal of Computational and Graphical Statistics*, 15(1):246–261, 2006.

[2] Anthony E Brockwell and Joseph B Kadane. Identification of regeneration times in MCMC simulation, with application to adaptive schemes. *Journal of Computational and Graphical Statistics*, 14(2), 2005.

[3] Jonathan MR Byrd, Stephen A Jarvis, and Abhir H Bhalerao. Reducing the runtime of MCMC programs by multithreading on SMP architectures. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8. IEEE, 2008.

[4] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis–Hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[5] Walter R Gilks, Gareth O Roberts, and Sujit K Sahu. Adaptive Markov Chain Monte Carlo through regeneration. *Journal of the American Statistical Association*, 93(443):1045–1054, 1998.

[6] Paolo Giordani and Robert Kohn. Adaptive independent Metropolis–Hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics*, 19(2):243–259, 2010.

[7] Guangbao Guo. Parallel statistical computing for statistical inference. *Journal of Statistical Theory and Practice*, 6(3):536–565, 2012.

[8] Lars Holden, Ragnar Hauge, and Marit Holden. Adaptive independent Metropolis–Hastings. *The Annals of Applied Probability*, pages 395–413, 2009.

[9] Guthrie Miller. Markov Chain Monte Carlo calculations allowing parallel processing using a variant of the metropolis algorithm. *Open Numer Methods J*, 2:12–7, 2010.

[10] Per Mykland, Luke Tierney, and Bin Yu. Regeneration in Markov chain samplers. *Journal of the American Statistical Association*, 90(429):233–241, 1995.

[11] Gareth O Roberts and Jeffrey S Rosenthal. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2009.

[12] Sheldon M. Ross. *Simulation, Fourth Edition*. Academic Press, Inc., Orlando, FL, USA, 2006.

[13] Antti Solonen, Pirkka Ollinaho, Marko Laine, Heikki Haario, Johanna Tamminen, and Heikki Järvinen. Efficient MCMC for climate model parameter estimation: Parallel adaptive chains and early rejection. *Bayesian Analysis*, 7(3):715–736, 2012.

[14] Ingvar Strid. Efficient parallelisation of Metropolis–Hastings algorithms using a prefetching approach. *Computational Statistics & Data Analysis*, 54(11):2814–2835, 2010.

[15] Ingvar Strid, Paolo Giordani, and Robert Kohn. Adaptive hybrid Metropolis-Hastings samplers for DSGE models. Technical report, SSE/EFI Working Paper Series in Economics and Finance, 2010.

[16] Douglas N VanDerwerken and Scott C Schmidler. Parallel Markov Chain Monte Carlo. *arXiv preprint arXiv:1312.7479*, 2013.

[17] Xiao-Lin Wu, Chuanyu Sun, Timothy M Beissinger, Guilherme JM Rosa, Kent A Weigel, Natalia de Leon Gatti, and Daniel Gianola. Parallel Markov chain Monte Carlo – bridging the gap to high-performance Bayesian computation in animal breeding and genetics. *Genetics Selection Evolution*, 44(1):29, 2012.