On Implementation of the Markov Chain Monte Carlo Stochastic Approximation Algorithm

Yihua Jiang, Peter Karcher and Yuedong Wang

Abstract The Markov Chain Monte Carlo Stochastic Approximation Algorithm (MCMCSAA) was developed to compute estimates of parameters with incomplete data. In theory this algorithm guarantees convergence to the expected fixed points. However, due to its flexibility and complexity, care needs to be taken for implementation in practice. In this paper we show that the performance of MCMCSAA depends on many factors such as the Markov chain Monte Carlo sample size, the step-size of the parameter update, the initial values and the choice of an approximation to the Hessian matrix. Good choices of these factors are crucial to the practical performance and our results provide practical guidelines for these choices. We propose a new adaptive and hybrid procedure which is stable and faster while maintaining the same theoretical properties.

1 Introduction

We often face incomplete data such as censored and/or truncated data in survival analysis, and unobservable latent variables in mixed effects and errors-in-variables models. It is difficult to compute estimates of parameters such as maximum likelihood estimates (MLE) with incomplete data because the likelihood usually involves intractable integrals. For example, the likelihood function of the generalized linear mixed models (GLMM) involves an integral with respect to the random ef-

Yihua Jiang
Capital One Financial Corp., 15000 Capital One Dr, Richmond, VA 23238 e-mail: Yihua.Jiang@capitalone.com

Peter Karcher
e-mail: peter.karcher@gmx.de

Yuedong Wang
Department of Statistics & Applied Probability, University of California, Santa Barbara, CA 93106
e-mail: yuedong@pstat.ucsb.edu

fects which usually does not have a closed form for non-Gaussian observations. Laplace approximation may lead to large bias and inconsistent estimates (Breslow and Clayton 1993, Lin and Breslow 1996, Jiang 1998). McCulloch (1997) modified the EM and the Newton-Raphson algorithms for fitting GLMMs with integrals approximated by Monte Carlo (MC) methods. A drawback of the MC approach is that the iterative scheme converges to a random variable rather than to the expected fixed point. As a consequence, it is difficult to decide the MC sample size and a stopping rule for such an iterative procedure. Booth and Hobert (1999) pointed out these potential problems and proposed an empirical approach to increase the MC sample size. However, it is unclear if their algorithm converges in theory.

MCMCSAA proposed by Gu and Kong (1998) and Gu and Kong (2000) for incomplete data is a major breakthrough which guarantees convergence to the expected fixed points in theory. MCMCSAA uses stochastic approximation (SA) to update the parameters and Markov chain Monte Carlo (MCMC) to approximate integrals at each iteration. In this article we investigate the performance of MCMCSAA using simulations. Our goal is to reveal some practical issues involved in the implementation of MCMCSAA . We find that MCMCSAA can be sensitive to the choices of (a) the initial values, (b) the MCMC sample size and step-size of the parameter update, and (c) a matrix whose expected value approximates the Hessian of the target function. We provide recommendations for implementation and propose a hybrid algorithm which is stable and considerably faster.

We review the MCMCSAA in Section 2. We conduct simulations to illustrate potential pitfalls in the implementation of the MCMCSAA in Section 3. We propose a hybrid algorithm in Section 4. Concluding remarks are made in Section 5.

2 Markov Chain Monte Carlo Stochastic Approximation Algorithms

In this section we briefly review MCMCSAA proposed by Gu and Kong (1998) and Gu and Kong (2000). See also Kushner and Yin (1997), Gu and Zhu (2001), Gu and Zhu (2002) and Lai (2003).

Suppose that we want to solve the following estimating equations

$$\mathbf{h}(\theta) = \mathbf{0}, \tag{1}$$

where θ is a vector of parameters and $\mathbf{h}(\theta)$ is a vector valued function that can be written as the expectation of a function $\mathbf{H}(\theta, \mathbf{e})$ with respect to a random vector $\mathbf{e}$ with density function $f_{\mathbf{e}}(\mathbf{e})$:

$$\mathbf{h}(\theta) = E_{\mathbf{e}}[\mathbf{H}(\theta, \mathbf{e})] = \int \mathbf{H}(\theta, \mathbf{e}) f_{\mathbf{e}}(\mathbf{e}) d\mathbf{e}. \tag{2}$$

In survival analysis, for example, data might be right censored. The score function can be written in form (2) (Satten 1996). Another example is a GLMM when the

random effects are considered as missing data. Integrating respect to the vector of random effects leads to form (2) (McCulloch 1997, Gu and Kong 1998, Zhu and Lee 2003).

To apply the MCMCSAA, one needs to find a matrix $I(\theta,\mathbf{e})$ such that $E_{\mathbf{e}}[I(\theta,\mathbf{e})] \approx \partial\mathbf{h}/\partial\theta$ in the neighborhood of the solution. We consider the following three different choices of $I(\theta,\mathbf{e})$:

$$\begin{aligned} I_1(\theta,\mathbf{e}) &= -\partial\mathbf{H}(\theta,\mathbf{e})/\partial\theta, \\ I_2(\theta,\mathbf{e}) &= -\partial\mathbf{H}(\theta,\mathbf{e})/\partial\theta - \mathbf{H}(\theta,\mathbf{e})\mathbf{H}(\theta,\mathbf{e})^T, \\ I_3(\theta,\mathbf{e}) &= -\partial\mathbf{H}(\theta,\mathbf{e})/\partial\theta - \mathbf{H}(\theta,\mathbf{e})\mathbf{H}(\theta,\mathbf{e})^T + \mathrm{E}_{\mathbf{e}}[\mathbf{H}(\theta,\mathbf{e})]\mathrm{E}_{\mathbf{e}}[\mathbf{H}(\theta,\mathbf{e})]^T. \end{aligned} \tag{3}$$

Whenever the meaning is clear we will drop the dependence on θ and $\mathbf{e}$ from all I-matrices. Also for brevity we will often refer to I_1, I_2, or I_3 when we actually mean the algorithm based on I_1, I_2, or I_3. I_1, used by Benveniste, Métivier and Priouret (1987), is usually well-behaved. For example, it is positive definite for convex target functions. Gu and Kong (2000) proposed I_2 and claimed that it is more efficient. I_3, motivated by the actual derivative of $\mathbf{h}(\theta)$ with respect to θ^T (Louis 1982), is new. Convergence of the algorithms are guaranteed for all three choices of I-matrices (Benveniste et al. 1987, Gu and Kong 1998, Gu and Kong 2000).

Let $\{\gamma_k, k \geq 1\}$ be a sequence of real numbers, and $\{m_k, k \geq 1\}$ be a sequence of positive integers which satisfy the following conditions

C1. $\gamma_k \leq 1$ for all k,

C2. $\sum_{k=1}^{\infty} \gamma_k = \infty$,

C3. $\sum_{k=1}^{\infty} \gamma_k^{1+\varepsilon}/m_k < \infty$, for some $\varepsilon \in (0,1)$,

C4. $\sum_{k=1}^{\infty} |\gamma_k/m_k - \gamma_{k-1}/m_{k-1}| < \infty$.

At iteration k, a MCMC sample of size m_k with equilibrium distribution $f_{\mathbf{e}}(\mathbf{e})$ is drawn. See Gilks, Richardson and Spiegelhalter (1996) for an overview of the Metropolis-Hastings algorithm. Let $\mathbf{e}_k^{(1)}, \ldots, \mathbf{e}_k^{(m_k)}$ be the MCMC sample of size m_k after some burn-in. The MCMCSAA updates the parameter vector θ and a matrix Γ as follows:

$$\begin{aligned} \Gamma_k &= (1-\gamma_k)\Gamma_{k-1} + \gamma_k \bar{I}_k, \\ \theta_k &= \theta_{k-1} + \Gamma_k^{-1}\bar{\mathbf{H}}_k, \end{aligned} \tag{4}$$

where

$$\bar{\mathbf{H}}_k = \sum_{j=1}^{m_k} \mathbf{H}(\theta_{k-1}, \mathbf{e}_k^{(j)})/m_k, \tag{5}$$

$$\bar{I}_k = \sum_{j=1}^{m_k} I(\theta_{k-1}, \mathbf{e}_k^{(j)})/m_k. \tag{6}$$

The I matrix in (6) may take any form in (3). Γ_k acts as a proxy of the Hessian matrix and is updated as parameters. γ_k is the step-size of the parameter updates. The innovation of MCMCSAA is the introduction of two sequences $\{\gamma_k\}$ and $\{m_k\}$. For a constant m_k and $\gamma_k = 1$, the estimate of θ will maintain a certain amount of variation along iterations due to estimating $\mathbf{H}$ and I by averaging over random samples. By increasing the sample size m_k, decreasing step-size γ_k, or both, the MCMCSAA decreases the variation in θ as iteration increases. Gu and Kong (2000) have shown that under some regularity conditions and when m_k and γ_k satisfy conditions (C1)-(C4), θ_k converges to the solution of (1) almost surely. The following are three combinations of m_k and γ_k that satisfy conditions (C1)-(C4):

G1. $\gamma_k = 1$ and $m_k = m_0 + k^2$,

G2. $\gamma_k = 1/k$ and $m_k = m_0$,

G3. $\gamma_k = 1/\sqrt{k}$ and $m_k = m_0 + k$,

where m_0 is the starting MCMC sample size. We see that when $\gamma_k = 1$, m_k needs to increase quadratically. When m_k is a constant, γ_k needs to decrease in the order of $1/k$. G3 is a compromise between G1 and G2.

The result of Gu and Kong (2000) provides the theoretical justification for the MCMCSAA. However, few research has been done to investigate its performance in practice. The simulation results in the next section are intended to show some potential problems that may occur in practice, and to compare different choices of parameters of the algorithm. We propose a new hybrid procedure and conduct more simulations in Section 4.

3 Simulations

To implement the MCMCSAA, one needs to decide (a) the initial values θ_0, (b) the MCMC sample size m_k and step-size of the parameter update γ_k, and (c) the form of the I-matrix. We will use simulations to show that the stability of the algorithm and the speed of convergence depends critically on all three choices.

We use the following simple model to illustrate challenges and potential pitfalls involved in the implementation of the MCMCSAA. Same issues arise in more complex models. Consider an experiment with $n = 10$ binary observations from each of $m = 20$ subjects. Binary observations are generate from the following simple GLMM:

$$\begin{aligned} &P(y_{ij} = 1|b_i) = \exp(b_i)/(1+\exp(b_i)), \\ &b_i \overset{iid}{\sim} \mathrm{N}(0,\theta), \;\; i = 1,\ldots,m; \;\; j = 1,\ldots,n, \end{aligned} \tag{7}$$

where b_i are random effects. Note that there is no fixed effects in model (7). The problems we are going to show in this section are usually associated with the esti-

mation of variance components such as θ, while the estimation of fixed effects is stable and converges quickly. Thus we set them to zero to simplify the exposition.

Let $\mathbf{y} = (y_{11}, \cdots, y_{1n}, \cdots, y_{m1}, \cdots, y_{mn})^T$ and $\mathbf{b} = (b_1, \cdots, b_m)$. The log-marginal distribution of $\mathbf{y}$ is

$$l_{\mathbf{y}}(\mathbf{y};\theta) = \log \int f_{\mathbf{y}|\mathbf{b}}(\mathbf{b};\theta) \exp\{l_{\mathbf{b}}(\mathbf{b};\theta)\} d\mathbf{b}, \tag{8}$$

where $f_{\mathbf{y}|\mathbf{b}}(\mathbf{b};\theta)$ is the conditional distribution of $\mathbf{y}$ conditional on $\mathbf{b}$ and $l_{\mathbf{b}}(\mathbf{b};\theta)$ is the log-density of $\mathbf{b}$. It can be shown that under some regularity conditions,

$$\frac{\partial l_{\mathbf{y}}(\mathbf{y},\theta)}{\partial \theta} = \int \frac{\partial l_{\mathbf{b}}(\mathbf{b};\theta)}{\partial \theta} f_{\mathbf{b}|\mathbf{y}}(\mathbf{b}) d\mathbf{b} = \mathrm{E}_{\mathbf{b}|\mathbf{y}} \frac{\partial l_{\mathbf{b}}(\mathbf{b};\theta)}{\partial \theta},$$

where $f_{\mathbf{b}|\mathbf{y}}$ is the posterior density of $\mathbf{b}$ given $\mathbf{y}$. Thus we can apply MCMCSAA with $\mathbf{e} = \mathbf{b}|\mathbf{y}$ and $\mathbf{H}(\theta,\mathbf{b}) = \partial l_{\mathbf{b}}(\mathbf{b},\theta)/\partial\theta$. It is easy to check that for model (7),

$$\begin{aligned}
\mathbf{H}(\theta,\mathbf{b}) &= -m/(2\theta) + \sum_{i=1}^{m} b_i^2/(2\theta^2), \\
I_1(\theta,\mathbf{b}) &= -m/(2\theta^2) + \sum_{i=1}^{m} b_i^2/\theta^3, \\
I_2(\theta,\mathbf{b}) &= I_1(\theta,\mathbf{b}) - \left[-m/(2\theta) + \sum_{i=1}^{m} b_i^2/(2\theta^2)\right]^2, \\
I_3(\theta,\mathbf{b}) &= I_2(\theta,\mathbf{b}) + (\mathrm{E}_{\mathbf{b}|\mathbf{y}}\mathbf{H}(\theta,\mathbf{b}))^2.
\end{aligned} \tag{9}$$

Since θ is non-negative, we set $\theta_k = \theta_{k-1}$ if the update $\theta_{k-1} + \gamma_k \Gamma_k^{-1} \bar{\mathbf{H}}_k < 0$.

We estimate $\mathrm{E}_{\mathbf{b}|\mathbf{y}}\mathbf{H}(\theta,\mathbf{b})$ by $\bar{H}(\theta,\mathbf{b}) = \sum_{j=1}^{m} \mathbf{H}(\theta,\mathbf{b}^{(j)})$, where $\mathbf{b}^{(j)}$ are MCMC samples. We use the single component Metropolis-Hastings algorithm to sample from $f_{\mathbf{b}|\mathbf{y}}(\mathbf{b})$ with the length of burn-in set to 300 (Gilks et al. 1996, Karcher and Wang 2002). Specifically, at the p^{th} iteration of the MCMC, we update each element $b_1^{(p)}, ..., b_m^{(p)}$ one by one as follows: generate X from $\mathrm{N}(b_i^{(p-1)}, 0.5\theta_{k-1})$ as a candidate for $b_i^{(p)}$ and U from Unif$(0,1)$; set $b_i^{(p)} = X$ if $U \le \min\{1, \pi(X)/\pi(b_i^{(p-1)})\}$ and $b_i^{(p)} = b_i^{(p-1)}$ otherwise.

We simulate data with three true values of θ: 0.5, 1.0 and 2.0. Denote θ_0 as the initial value and θ_{MLE} as the MLE. We compute θ_{MLE}'s numerically by maximizing (8) using integration and optimization functions in R (www.r-project.org). We consider the following combinations of parameters for the algorithm:

- Initial values θ_0: $.5\theta_{MLE}$, θ_{MLE} and $2\theta_{MLE}$,
- I-matrices: I_1, I_2 and I_3,
- γ_k and m_k: G1, G2 and G3,
- m_0: 30 and 300.

We have also tried to set both m_0 and burn-in period to 10000, the convergence behavior remains the same. For each combination of parameters, we repeat simulation 100 times. Following Gu and Kong (2000), we stop the algorithm after 50 iterations for G1, 1000 iterations for G2, and 250 iterations for G3.

We first look at the convergence behavior. All iteration sequences follow one of the following four patterns: (a) converged to the MLE, (d) approaching but not yet converged to the MLE, (c) converges to zero, and (d) diverges to infinity. We show a typical example for each pattern in Figure 1.

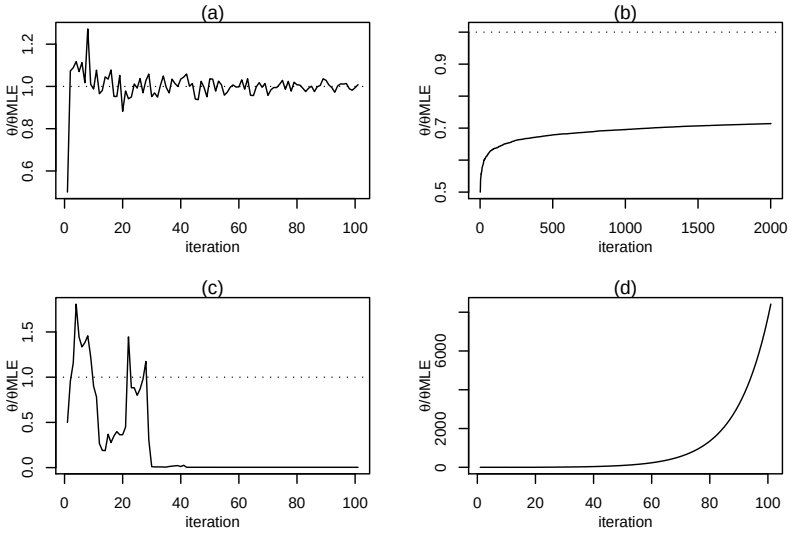

Fig. 1 Four patterns of sequences divided by θ_{MLE}.

The curve of sequences usually becomes smoother and smoother along the iterations. However, depending on the choice of γ_k and m_k, random variation persists even after a large number of iterations. The following definitions allow the estimates to have small variations around the MLE. Let $\bar{\theta}_5$ be average of the estimate at the last 5 iterations. For any sequence, we define the following three states:

(S1) *converged*, if $\left|(\bar{\theta}_5 - \theta_{MLE})/(\theta_{MLE}+1)\right| < 0.05$,

(S2) *diverged*, if $\left|(\bar{\theta}_5 - \theta_{MLE})/(\theta_{MLE}+1)\right| > 1$, or $\left|(\bar{\theta}_5 - \theta_{MLE})/(\theta_{MLE}+1)\right| \geq 0.05$ and $\left|\bar{\theta}_5/\theta_{MLE}\right| < 0.05$,

(S3) *not converged*, otherwise.

Not converged sequences (S3) consists of two classes: sequences that are converging but are not close enough to the MLE and sequences that are diverging but are not far enough from the MLE. Tables 1 shows the convergence behavior. An entry i/j indicates that there are i converged sequences, j diverged sequences, and $100-i-j$ not converged sequences.

θ	θ_0		$m_0 = 30$ G1	G2	G3	$m_0 = 300$ G1	G2	G3
		$\mathbf{I}_1$	78/0	0/0	72/0	78/0	0/0	75/0
	$0.5\theta_{MLE}$	$\mathbf{I}_2$	8/85	55/26	43/51	11/86	74/17	61/32
		$\mathbf{I}_3$	38/47	53/6	70/15	50/38	80/6	75/15
		$\mathbf{I}_1$	79/0	64/0	85/0	83/0	92/0	89/0
0.5	$1\theta_{MLE}$	$\mathbf{I}_2$	5/93	55/36	32/64	21/75	84/12	62/30
		$\mathbf{I}_3$	38/54	65/8	60/18	53/36	85/9	71/14
		$\mathbf{I}_1$	82/0	46/0	87/0	87/0	55/0	90/0
	$1.5\theta_{MLE}$	$\mathbf{I}_2$	4/95	27/65	16/80	11/88	46/47	24/71
		$\mathbf{I}_3$	34/54	63/9	50/36	50/44	83/5	71/22
		$\mathbf{I}_1$	95/0	0/0	94/0	97/0	0/0	94/0
	$0.5\theta_{MLE}$	$\mathbf{I}_2$	10/85	65/22	49/44	22/77	77/17	72/24
		$\mathbf{I}_3$	79/19	75/3	83/6	79/17	97/1	92/2
		$\mathbf{I}_1$	97/0	86/0	98/0	95/0	100/0	99/0
1	$1\theta_{MLE}$	$\mathbf{I}_2$	9/87	66/29	58/38	51/48	96/3	89/9
		$\mathbf{I}_3$	64/33	81/6	81/7	81/18	98/0	94/6
		$\mathbf{I}_1$	96/0	77/0	98/0	97/0	89/0	98/0
	$1.5\theta_{MLE}$	$\mathbf{I}_2$	3/95	28/64	26/72	6/93	38/54	30/69
		$\mathbf{I}_3$	67/31	82/6	86/9	80/19	89/3	91/7
		$\mathbf{I}_1$	98/0	0/0	98/0	98/0	0/0	99/0
	$0.5\theta_{MLE}$	$\mathbf{I}_2$	11/88	66/24	58/40	32/66	53/42	49/49
		$\mathbf{I}_3$	84/14	95/1	97/1	91/9	100/0	100/0
		$\mathbf{I}_1$	98/0	97/0	99/0	99/0	100/0	100/0
2	$1\theta_{MLE}$	$\mathbf{I}_2$	12/87	85/13	58/40	58/42	99/1	95/4
		$\mathbf{I}_3$	85/15	96/0	93/5	91/8	100/0	99/1
		$\mathbf{I}_1$	98/0	87/0	99/0	100/0	98/0	100/0
	$1.5\theta_{MLE}$	$\mathbf{I}_2$	8/91	35/64	19/81	5/95	16/84	12/88
		$\mathbf{I}_3$	82/17	78/9	91/8	92/5	82/3	98/0

Table 1 Number of simulation replications that are converged or diverged.

It is clear that the convergence behavior depends on θ_0 as well as the choices of I, γ_k and m_k. As always, a good initial value is important. I_1 is stable as expected: most all sequences based on I_1 converged and none of them diverged. A closer look indicates that all not converged sequences are actually converging slowly. In contrast to I_1, I_2 and I_3 are unstable with many sequences diverged, regardless of the choice of θ_0, I, γ_k and m_k. It is important to note that the problem persists even when the initial value is set to the target value θ_{MLE}. I_2 has more diverged sequences than I_3. Convergence also depends on the specific combination of I_1, I_2, I_3 and G1, G2, G3. For example, I_2 always performs best with G2. And when θ_0 is not close to θ_{MLE}, I_1 performs better with G1 than with G2.

To illustrate the cause of divergence when I_2 and I_3 are used, for a typical diverged case, we plot θ and Γ for the first 10 iterations in Figure 2. We see that from iteration 7, due to MCMC simulation error, the Γ values are negative (i.e. the Γ matrices are not positive definite). This turns θ away from the MLE. In most cases sequences with such problems do not recover and diverge to infinity or con-

verge to zero. Zhu and Lee (2003) noticed the same problem and proposed a hybrid algorithm.

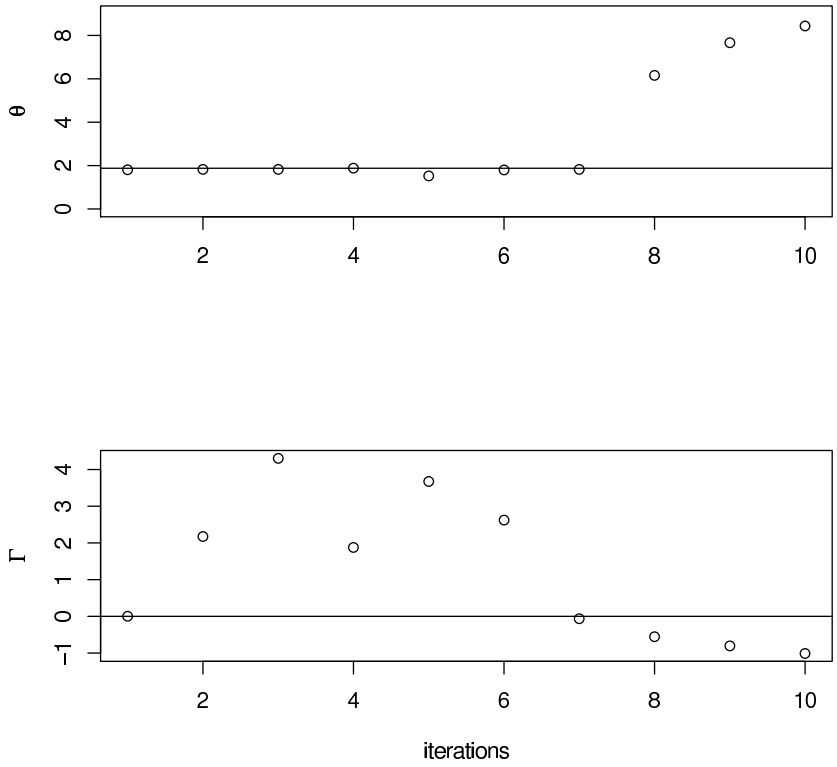

Fig. 2 Effect of negative Γ on estimate of θ. The bottom plot shows the value of Γ based on I_2. The top plot shows the estimate of θ at the corresponding iteration, and the solid line represents the MLE.

Next, we look at the rate of convergence. In Table 1 we see that for all combinations, the algorithm based on G2 has the most sequences that are classified as *not converged*. To compare G1 and G2, we show two typical sequences of estimates based on I_1 with $\theta_0 = \theta_{MLE}$ and $\theta_0 = 0.5\theta_{MLE}$ in Figure 3.

The left plot shows that θ/θ_{MLE} are close to 1 for both G1 ad G2, which means that both sequences settle around the MLE, but the curve based on G2 is much smoother than the curve based on G1. The right plot shows that the sequences based on G1 converged to the MLE in fewer than 50 iterations, whereas the sequence based on G2 has not reached the level of the MLE after 1000 iterations.

Plots under other settings are similar. We conclude that if the starting value is far from the MLE, G1 reaches the level of the MLE faster than G2. On the other hand if the estimate is already at the level of the MLE, G2 settles down much faster than G1. These results are not surprising since for G2 the step-size of the parameter

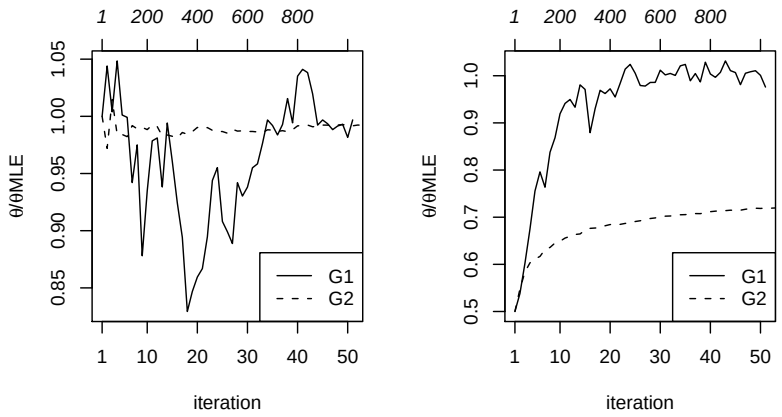

Fig. 3 Ratios between the estimates and θ_{MLE} for sequences based on G1 and G2. The iteration numbers of G1 are marked at the bottom of the x-axis and iteration numbers of G2 are marked at the top.

update γ_k is $1/k$. Thus after only a few iteration γ_k is very small and does not allow a large change in θ anymore. It takes many iterations to reach the level of the MLE. For G1, γ_k is always 1. Therefore, it can reach the level of the MLE quickly, but remain to have random variations after the level of the MLE has been reached. G3 is a compromise between G1 and G2 (not shown).

Above simulation results illustrate advantages and disadvantages of different combinations of the I matrices, step size γ_k and MCMC sample size m_k. These results can be used as a guide to make an algorithm stable and efficient. Overall, we recommend I_1 for stability. One may use G1 and then switch to G2 after the sequence has reached the target level (Section 4). When I_2 or I_3 are used, one needs to check the properties of the estimates, for example, if the Γ matrices are positive definite. One can start with I_1 and switch to I_2 or I_3 after a number of iterations (Zhu and Lee 2003). This would combine the stability of I_1 in the beginning of the algorithm with the faster convergence of I_2 or I_3 at the end of the algorithm.

4 A Hybrid Algorithm

We consider I_1 only in this section. Based on simulations in the previous section, we found that G1 reaches the MLE quickly and G2 reduces variation in the estimates quickly after the sequence reached the level of the MLE. Therefore, we consider the following class of algorithms:

$$\gamma_k = k^{-t_k} \text{ and } m_k = m_0 + k^{2(1-t_k)}, \quad 0 \le t_k \le 1. \tag{10}$$

Special cases include G1 with $t_k = 0$, G3 with $t_k = 0.5$, and G2 with $t_k = 1$. Note that the sequences need to satisfy conditions (C1)-(C4).

We would like t_k to be close to zero for small k and close to one for large k. The idea of our procedures is to select γ_k and m_k adaptively according to the actual convergence behavior. There could be many adaptive schemes. We consider a simple approach by fitting a regression line through the last K estimates of θ. If the slope is zero, it is likely that the estimates have reached the level of the MLE and the variation of θ around the regression line is likely due to MCMC sampling. In this case we want to reduce the variation using a combination of γ_k and m_k similar to G2. If the slope is not zero, i.e. the sequence is still increasing or decreasing, it is desirable to choose γ_k and m_k such that the estimates reach the level of MLE quickly. One such choice is G1. Let r_k be the correlation coefficient between the estimate of θ at the previous K iterations and their corresponding iteration numbers, and $\hat{r}_k$ be the sample correlation coefficient. Zero r_k corresponds to the zero slope of the regression line. We use t-test to test hypothesis $H_0 : r_k = 0$ *v.s.* $H_1 : r_k \neq 0$. Specifically, H_0 is rejected when $\left|\hat{r}_k / \sqrt{(1-\hat{r}_k^2)/(K-2)}\right| \geq t_{\alpha/2,K-2}$ where α is the significant level and $t_{\alpha/2,K-2}$ is the $1-\alpha/2$ percentile of the t-distribution with $K-2$ degrees of freedom. This suggests the following algorithms:

G4. For $k > K$, $t_k = 1 - r_k^2$,

G5. For $k > K$, $t_k = (1-r_k^2) I(|\hat{r}_k / \sqrt{(1-\hat{r}_k^2)/(K-2)}| < t_{\alpha/2,K-2})$,

G6. For $k > K$, $t_k = I(|\hat{r}_k / \sqrt{(1-\hat{r}_k^2)/(K-2)}| < t_{\alpha/2,K-2})$.

G1 is used for the first K iterations in G4, G5 and G6. To implement algorithms G4, G5 and G6, we need to decide the number K, that is, the last K estimates to be used to compute the correlation coefficients.

For the same simulated data we apply algorithms G4, G5 and G6 with four different choices of K: 10, 20, 30 and 40. We stop the hybrid algorithms after the 50^{th} iteration which is consistent with the number used for G1 in the previous section. Table 2 presents the number of converged sequences out of 100 simulation replications. There is no diverged sequences.

Table 2 indicates that the convergence behavior is insensitive to the choice of K so long it is large enough (≥ 10). To compare the convergence rate with G1, we plot a typical replication of hybrid algorithms with $K = 20$ in Figure 4. We see that the curves based on hybrid algorithms are much smoother than those based on G1. However, it is unclear whether 50 iterations are enough for the sequences to reach the θ_{MLE}. We need to consider stopping rules.

Different convergence criteria have been used in practice. One simple rule is to stop the algorithm when $|\theta_{k+1} - \theta_k| < \delta$ where δ is a preassigned number. Gu and Zhu (2001) proposed a two-stage algorithm where the first stage stops at K_1 where $K_1 = inf\left\{K \geq K_0 : ||\sum_{k=K-K_0+1}^{K} \frac{sgn(\theta_k - \theta_{k-1})}{K_0}|| \leq \delta_1\right\}$, K_0 and δ_1 are preassigned constants and sgn is the signum function of a real number x which returns 1 when $x > 0$, 0 when $x = 0$ and -1 when $x < 0$. The second stage utilizes the trace of Γ matrix and H matrix to decide the stopping rule. We use the variance along

		True value θ								
		0.50			1.00			2.00		
θ_0		G4	G5	G6	G4	G5	G6	G4	G5	G6
$0.5\theta_{MLE}$	$K=10$	64	76	73	90	94	96	96	98	99
	$K=20$	65	78	75	89	94	97	97	97	98
	$K=30$	72	78	76	89	94	95	97	98	100
	$K=40$	69	75	77	93	95	96	98	99	97
$1\theta_{MLE}$	$K=10$	83	83	84	90	95	95	99	98	99
	$K=20$	73	80	83	93	96	94	97	100	98
	$K=30$	73	82	81	91	93	96	100	99	100
	$K=40$	82	85	79	96	94	96	99	99	99
$1.5\theta_{MLE}$	$K=10$	86	84	79	92	93	99	97	99	99
	$K=20$	75	86	87	91	96	98	97	98	99
	$K=30$	75	87	86	96	93	99	99	99	100
	$K=40$	84	83	84	93	94	95	100	99	98

Table 2 Number of simulation replications that are converged.

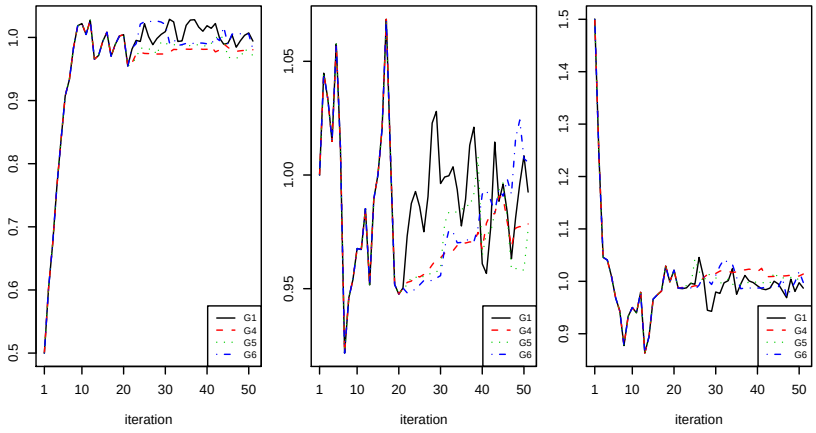

Fig. 4 Ratios between the estimates and θ_{MLE} for sequences based on G1, G4, G5 and G6.

.

iterations to measure whether the sequence is stable or not. Specifically we consider the following two stopping rules:

I. At the k^{th} iteration, stop if $\frac{|\theta_k-\theta_{k-1}|}{\sqrt{v}+\delta_1} < \delta_2$ where δ_1, δ_2 are predetermined numbers and v is the sample variance of $\{\theta_0,\ldots\theta_k\}$;

II. Since Γ_k^{-1} is an estimate of the covariance matrix of $\hat{\theta}$, therefore, instead of v, we stop if $\frac{|\theta_k-\theta_{k-1}|}{\sqrt{\Gamma_k^{-1}}+\delta_1} < \delta_2$.

For the same simulation settings, we apply algorithms G1-G6 using above two stopping rules. For both rules, we set $\delta_1 = 0.001$ and $\delta_2 = 0.001$. We compute k as the

average of number of iterations needed to satisfy the stopping criterion, *conv.* as converged cases after 600 iterations, *diff* as the average of the difference between estimates and the θ_{MLE} among converged sequences, and *CPU* as the average CPU time in seconds for one simulation replication.

Tables 3 and 4 show convergence behavior under rules I and II respectively. We did not include the convergence behavior of G2 because G2 performs poorly under these two stopping rules. G2 makes the algorithm stop early even though the iteration is far away from the targeted θ_{MLE}.

θ	θ_0		G1	G4	G5	G6	G3
0.5	$0.5\theta_{MLE}$	k	118.59	50.12	55.02	53.53	77.40
		conv.	87	68	78	78	47
		diff	6.672	19.60	12.55	14.99	38.5
		CPU (s)	66.12	1.34	4.38	4.67	2.64
0.5	$1\theta_{MLE}$	k	216.40	80.38	82.44	71.86	179.42
		conv.	92	82	83	84	84
		diff	2.677	10.66	8.424	9.49	7.407
		CPU (s)	390.82	2.71	16.49	9.80	6.77
0.5	$1.5\theta_{MLE}$	k	131.41	45.04	53.75	51.84	87.12
		conv.	92	73	78	84	70
		diff	5.881	13.90	11.46	10.16	15.23
		CPU (s)	113.02	0.94	4.09	4.10	3.04
1	$0.5\theta_{MLE}$	k	125.4	45.8	49.49	43.13	66
		conv.	95	87	95	91	65
		diff	7.588	22.13	14.15	18.86	43.98
		CPU (s)	84.55	1.03	2.70	2.03	2.25
1	$1\theta_{MLE}$	k	289.66	90.9	89.66	77.59	222.02
		conv.	98	94	97	98	99
		diff	3.188	12.41	9.512	11.95	10.35
		CPU (s)	828.94	3.03	17.29	10.797	8.9
1	$1.5\theta_{MLE}$	k	156.46	90.9	56.05	49.42	88.16
		conv.	98	94	95	93	89
		diff	6.854	20.41	17.67	17.10	20.92
		CPU (s)	199.19	1.11	4.56	3.38	3.11
2	$0.5\theta_{MLE}$	k	132.87	44.82	49.33	43.01	68.16
		conv.	100	97	96	98	84
		diff	13.46	43.17	34.76	31.84	57.68
		CPU (s)	111.41	0.93	2.92	1.88	2.31
2	$1\theta_{MLE}$	k	272.06	96.15	90.54	79.69	239.70
		conv.	99	97	100	100	94
		diff	8.389	26.86	17.34	17.89	20.89
		CPU (s)	720.95	3.27	14.30	11.57	9.47
2	$1.5\theta_{MLE}$	k	162.57	49.47	55.96	47.83	80.69
		conv.	100	94	97	99	94
		diff	15.68	40.31	30.07	30.78	37.05
		CPU (s)	195.29	1.07	4.44	2.99	2.80

Table 3 Convergence behavior of G1, G3, G4, G5, and G6 under stopping rule I, *diff* are multiplied by 1000 for ease of presentation.

θ	θ_0		G1	G4	G5	G6	G3
0.5	$0.5\theta_{MLE}$	k	72.58	38.59	43.95	39.79	43.32
		conv.	83	59	70	70	24
		diff	9.562	23.15	17.69	18.51	59.47
		CPU (s)	14.02	1.55	3.81	2.55	1.40
0.5	$1\theta_{MLE}$	k	78.08	32.16	34.53	32.79	35.41
		conv.	88	72	75	77	73
		diff	7.743	16.12	15.23	13.35	14.72
		CPU (s)	16.28	1.20	1.76	1.68	1.16
0.5	$1.5\theta_{MLE}$	k	68.98	32	35.08	34.32	38.89
		conv.	88	74	77	76	57
		diff	8.929	16.53	13.36	14.36	26.09
		CPU (s)	11.97	1.18	1.85	1.77	1.27
1	$0.5\theta_{MLE}$	k	71.37	33.06	38.46	36.41	38.48
		conv.	96	81	91	91	41
		diff	11.90	28.44	21.90	24.23	66.73
		CPU (s)	13.83	1.26	2.17	1.96	1.24
1	$1\theta_{MLE}$	k	63.86	30.2	33.91	32.68	35.98
		conv.	94	87	91	89	92
		diff	11.16	23.01	18.18	20.98	19.49
		CPU (s)	10.07	1.12	1.68	1.61	1.17
1	$1.5\theta_{MLE}$	k	65.60	30.88	31.86	33.01	36.02
		conv.	98	92	91	88	84
		diff	10.75	21.11	21.80	24.16	27.19
		CPU (s)	12.44	1.14	1.48	1.64	1.17
2	$0.5\theta_{MLE}$	k	67.01	34.21	34.19	32.87	35.21
		conv.	98	94	96	97	61
		diff	22.34	48.81	43.22	42.02	13.57
		CPU (s)	11.25	1.28	1.66	1.62	1.14
2	$1\theta_{MLE}$	k	65.75	29.83	29.85	27.5	32.81
		conv.	100	96	96	97	95
		diff	21.46	47.51	39.91	48.30	37.52
		CPU (s)	10.89	1.10	1.30	1.13	1.07
2	$1.5\theta_{MLE}$	k	68.42	29.67	32.35	30.73	33.84
		conv.	100	95	99	96	99
		diff	23.19	50.40	41.64	47.49	51.93
		CPU (s)	12.85	1.10	1.53	1.38	1.10

Table 4 Convergence behavior of G1, G3, G4, G5 and G6 under stopping rule II, *diff* are multiplied by 1000 for ease of presentation.

First, we see that stopping rule I is much more time-consuming than stopping rule II for G1, but not for the hybrid algorithms. And when $\theta_0 = \theta_{MLE}$, G1 with stopping rule I is very slow to converge, the average CPU time is almost 200 times of those for hybrid algorithms.

Overall, G1 has the highest *converged* cases among all the algorithms. On the other hand, G1 is always the most time-consuming one. G3 usually has the lowest *converged* cases especially when θ_0 is far away from the θ_{MLE}. G3 is the tradeoff between G1 and G2 and the sequences based on G3 is much smoother than G1. So if we use the variation of sequences as the stopping rule, G3 may stop prematurely. The hybrid algorithms overcome this problem by switching between G1 and G2,

so they give better *converged* rate than G3 even though the average CPU time for hybrid algorithms and G3 are at the same level.

Now we look at the average difference between estimates and θ_{MLE}. Under stopping rule I, when $\theta = 0.5$ and $\theta = 1$, the average difference is at 10^{-3} level for G1, and at 10^{-2} level for hybrid algorithms; when $\theta = 2$, the average difference is at 10^{-2} level for both G1 and hybrid algorithms. Under stopping rule II, when $\theta = 0.5$, the average difference is at 10^{-3} level for G1, and at 10^{-2} level for hybrid algorithms; when $\theta = 1$ and $\theta = 2$, the average difference is at 10^{-2} level for both G1 and hybrid algorithms. The average difference based on G1 are always smaller then the hybrid algorithms. However, the CPU time for G1 is between 10 to 200 times of that for the hybrid algorithms.

5 Conclusions

MCMCSAA is a powerful tool to compute estimates of parameters with incomplete data. It has attractive theoretical properties. However, due to its complexity and flexibility, the implementation of MCMCSAA is far from straightforward and caution needs to be exercised.

The MCMCSAA involves several parameters including the form of the I-matrix, the MCMC sample size m_k and step-size of the parameter update γ_k. We have shown that the performance of MCMCSAA depends critically on all three parameters. We recommend I_1 over I_2 and I_3 for stability. When I_2 or I_3 is used, one should check the property of the H matrix to avoid divergence. There can be many choices for the combination of the MCMC sample size m_k and step-size of the parameter update γ_k. Different choices represent different compromise between convergence rate and CPU time. There is no ultimate best algorithm for both precision and efficiency. Therefore, different choice should be chosen for different purpose. In general, the proposed hybrid algorithms are stable and fast.

References

Benveniste, A., Métivier, M. and Priouret, B. (1987). *Adaptive Algorithms and Stochastic Approximations: Theory and Applications of Signal Detection and Processing and Pattern Recognition (French)*, Masson:Pari:NY.

Booth, J. and Hobert, J. (1999). Maximizing generalized linear mixed model likelihoods with an automated monte carlo em algorithm, *Journal of the Royal Statistical Society B* **61**: 265–285.

Breslow, N. E. and Clayton, D. G. (1993). Approximate inference in generalized linear mixed models, *Journal of the American Statistical Association* **88**: 9–25.

Gilks, W. R., Richardson, S. and Spiegelhalter, D. J. E. (1996). *Markov Chain Monte Carlo in Practice*, Chapman and Hall:London.

Gu, M. G. and Kong, F. H. (1998). A stochastic approximation algorithm with markov chain monte-carlo for imcomplete data estimation problems, *Proceedings of the National Academy of Sciences* **95**: 7270–7274.

Gu, M. G. and Kong, F. H. (2000). A generalized markov chain monte carlo stochastic approximation algorithm for statistical computing, *Personal communication*.

Gu, M. G. and Zhu, H. T. (2001). Maximum likelihood estimation for spatial models by markov chain monte carlo stochastic approximation, *Journal of the Royal Statistical Society B* **63**: 339–355.

Gu, M. G. and Zhu, H. T. (2002). Maximum likelihood estimation for generalized random effect models, *Personal communication*.

Jiang, J. (1998). Consistent estimators in generalized linear mixed models, *Journal of the American Statistical Association* **93**: 720–729.

Karcher, P. and Wang, Y. (2002). Generalized nonparametric mixed effects models, *Journal of Computational and Graphical Statistics* **10**: 641–655.

Kushner, H. J. and Yin, G. G. (1997). *Stochastic Approximations Algorithms and Applications*, Springer, New York.

Lai, T. L. (2003). Stochastic approximation, *Annals of Statistics* pp. 391–406.

Lin, X. and Breslow, N. E. (1996). Bias correction in generalized linear mixed models with multiple components of dispersion, *Journal of the American Statistical Association* **91**: 1007–1016.

Louis, T. A. (1982). Finding the observed information matrix when using the Em algorithm, *Journal of the Royal Statistical Society B* **44**: 226–233.

McCulloch, C. (1997). Maximum likelihood algorithms for generalized linear mixed models, *Journal of the American Statistical Association* **92**: 162–170.

Satten, G. A. (1996). Rank-based inference in the proportional hazards model for interval censored data, *Biomtrika* **83**: 355–370.

Zhu, H. T. and Lee, S. Y. (2003). Analysis of generalized linear mixed models via a stochastic approximation algorithm with markov chain Monte-Carlo method, *Statistics and Computing* pp. 391–406.