RMCMC: A system for updating Bayesian models

CrossMark

F. Din-Houn Lau *, Axel Gandy

Department of Mathematics, Imperial College London, South Kensington Campus, London, SW7 2AZ, UK

A B S T R A C T

A system to update estimates from a sequence of probability distributions is presented. The aim of the system is to quickly produce estimates with a user-specified bound on the Monte Carlo error. The estimates are based upon weighted samples stored in a database. The stored samples are maintained such that the accuracy of the estimates and quality of the samples are satisfactory. This maintenance involves varying the number of samples in the database and updating their weights. New samples are generated, when required, by a Markov chain Monte Carlo algorithm. The system is demonstrated using a football league model that is used to predict the end of season table. The correctness of the estimates and their accuracy are shown in a simulation using a linear Gaussian model.

© 2014 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/3.0/).

1. Introduction

We are interested in producing estimates from a sequence of probability distributions. The aim is to quickly report these estimates with a user-specified bound on the Monte Carlo error. We assume that it is possible to use MCMC methods to draw samples from the target distributions. For example, the sequence can be the posterior distributions of parameters from a Bayesian model as additional data becomes available, with the aim of reporting the posterior means with the variance of the Monte Carlo error being less than 0.01. We present a general system that addresses this problem.

Our system involves saving the samples produced from the MCMC sampler in a database. The samples are updated each time there is a change of sample space. The update involves weighting or transiting the samples, depending on whether the space sample changes or not. In order to control the accuracy of the estimates, the samples in the database are maintained. This maintenance involves increasing or decreasing the number of samples in the database. This maintenance also involves monitoring the quality of the samples using their effective sample size. See Table 1 for a summary of the control variables. Another feature of our system is that the MCMC sampler is paused whenever the estimate is accurate enough. The MCMC sampler can later be resumed if a more accurate estimate is required. Therefore, it may be the case that no new samples are generated for some targets. Hence the system is efficient, as it reuses samples and only generates new samples when necessary.

Our approach has similar steps to those used in sequential Monte Carlo (SMC) methods (Doucet et al., 2001; Liu, 2008), such as an update (or transition) step and re-weighting of the samples. Despite the similarities, SMC methods are unable to achieve the desired aims considered in this paper. Specifically, even though SMC methods are able to produce estimates from a sequence of distributions, it is unclear how to control the accuracy of this estimate without restarting the whole procedure. For example, consider the simulations in Gordon et al. (1993) where the bootstrap particle filter, a particular SMC method, is introduced. In these simulations the posterior mean is reported with the interval between 2.5 and 97.5 percentile points. As these percentile points are fixed, there is no way to reduce the length of the interval. The only hope of reducing the interval

---

* Corresponding author. Tel.: +44 7794463713.
   E-mail addresses: fdl06@ic.ac.uk, din-houn.lau06@imperial.ac.uk (F.D.-H. Lau).

Table 1
Summary of control variables.

Control variable	Measurement	Target interval
Accuracy of estimates (A)	Standard deviation of estimates	$[\beta_1, \beta_2]$
Quality of samples (Q)	Effective sample size of database/N_{MAX}	$[\gamma_1, \gamma_2]$

is to rerun the particle filter with more particles, although there is no guarantee. This conflicts with the aim of reporting the estimates quickly. In practice, most SMC methods are concerned with models where only one observation is revealed at a time (see simulations in e.g. Kitagawa, 2014, Del Moral et al., 2006 and Chopin, 2002). Our framework allows for observations to be revealed in batches of varying sizes; see the application presented in Section 3.

A potential application of the system is monitoring the performance of multiple hospitals where the data observed are patient records and the estimate of interest relates to the quality of patient care at each hospital. Controlling the accuracy of this estimate may relate to controlling the proportion of falsely inspected hospitals. Another example of a realistic application of the system is a football league model (see Section 3) where the data revealed are the match results and the estimate of interest is the end of season rank league table. Controlling the estimated rank may be of interest to sports pundits and gamblers.

In Section 2 we define, in detail, the setup we are considering. We then describe the separate processes of the system. We also describe how to combine the weighted samples to form the estimate of interest. Then in Section 2.7 we present a modified batch mean approach that we use to compute the accuracy of the estimate. In Section 3 we investigate the performance of the system using a model for a football league. For this application, the aim is to provide quick and accurate predictions of the end of season team ranks as football match results are revealed. We examine the performance of the system as the size of data received increases. Currently, there is no theoretical proof that the proposed system is stable; however simulations verify the correctness of the reported accuracy and the estimate. We present such a simulation in Section 4, where we apply the system using a linear Gaussian model and simulated data. We conclude in Section 5 with a discussion of potential future topics of research.

2. Description of the system

2.1. Setup

We now describe the settings we consider and the necessary operations required for our system to function. Let $(S_n, \mathscr{S}_n, \pi_n)_{n \in \mathbb{N}}$ be a sequence of probability spaces. We are interested in reporting $\pi_n g_n = \int g_n(x) d\pi_n(x)$ where g_n is a, possibly multivariate, random variable on $(S_n, \mathscr{S}_n, \pi_n)$. In order to implement our system, the following operations are required:

1. *MCMC*: for all $n \in \mathbb{N}$, generate samples from an MCMC with stationary distribution π_n.
2. *Weighting Samples*: for all $k \in \overline{D} := \{j : S_{j-1} = S_j \text{ and } \pi_j \ll \pi_{j-1}\}$, the Radon–Nikodym derivative $\frac{d\pi_j}{d\pi_{j-1}}$ can be evaluated.
3. *Transiting Samples*: for all $k \in D \, \exists f_k : S_{k-1} \times [0, 1] \to S_k$ such that $\xi \sim \pi_{k-1}, U \sim U[0, 1] \implies f_k(\xi, U) \sim \pi_k$.

The weighting operation enables us to weight previously generated samples according to the latest measure. In the case where the sample space changes or the Radon–Nikodym derivative is not defined, the transition operation allows us to map the samples to the latest measure. If such a transition function, in operation 3, is unavailable, then the following may be used instead.

3′. *Transiting Samples*: for all $k \in D \, \exists f_k : S_{k-1} \times [0, 1] \to S_k$.

This alternative transition operation allows us to map the samples into the latest sample space of interest.

2.2. Global variables

The samples produced by the RMCMC process (Section 2.3) are stored in a database. Each sample is recorded to the database with a production date and an information cut-off. The production date is the date and time the sample was written to the database from the RMCMC process. The information cut-off refers to the measure the MCMC was targeting when the sample was produced. Lastly, each sample will enter the database with weight 1. The maximum number of samples allowed in the database is N_{MAX}. In Section 2.5 we explain how the control process varies N_{MAX} over time. Further, we shall refer to the current number of samples in the database as N. The deletion process (Section 2.6) ensures that $N \leq N_{MAX}$ by sometimes removing samples from the database. A summary of the systems global variables is provided in Table 2 along with their descriptions.

2.3. RMCMC process

The RMCMC process, summarised in Algorithm 2.1, is an MCMC method that changes its target without the need to restart. When the target of interest changes from π_{j-1} to π_j so does the target of the MCMC. The Markov chain continues from the latest sample, making a transition step using $\hat{f}_j$ if there is a change of sample space. This ensures that the next MCMC is

---

Algorithm 2.1 RMCMC process

   Parameters: MCMC algorithm, B_0.

 1: **repeat** indefinitely.
 2:     **if** target changes to π_j **then**
 3:         Set $B = B_0$.
 4:         Update target of MCMC to π_j.
 5:         **if** $j \in D$ **then**
 6:             Set current position of MCMC to $f_j(\xi, U)$ where $U \sim U[0, 1]$ and ξ is the latest sample generated.
 7:     **if** RMCMC_ON=TRUE **then**
 8:         Perform MCMC step.
 9:         **if** $B = 0$ **then** write sample to database with weight 1.
10:         **else** $B \leftarrow B - 1$.
11:     **else** sleep for some time.

---

exploring the correct space. We continue from this sample in the hope that the Markov chain converges faster to the updated target distribution than a randomly chosen starting point. To allow the Markov chain to move toward the updated target distribution we use a burn-in period where the first B_0 samples are discarded after the target changes. This burn-in period will also weaken the dependence between samples from different target distributions. As this MCMC method is never reset and continues from the last generated sample we refer to it as a rolling MCMC (RMCMC) process.

The RMCMC process is only active when new samples are required as it can be paused and resumed by the control process (Section 2.5). If the process is paused for long periods, it may be the case that no samples are produced for some targets. In Algorithm 2.1 the generated samples are written to the database individually. In practice, however, it may be more convenient to write the samples to the database in batches. This practice is allowable and will not affect the functioning of the system.

2.4. Update process

The update process, presented in Algorithm 2.2, ensures that the samples are weighted correctly each time the target changes. There are two types of updates depending on the measures and their sample spaces. More precisely, consider a change of target from π_{j-1} to π_j. If $j \in D$, that is the sample spaces differ or the Radon–Nikodym $d\pi_j/d\pi_{j-1}$ is not defined, then the function f_k is used to map the samples in the database onto the new space. On the other hand, if $j \notin D$, the samples are first re-weighted according to $d\pi_j/d\pi_{j-1}$, then scaled. We now discuss these re-weight and scaling steps in more detail.

Algorithm 2.2 Update Process

 1: **repeat** indefinitely.
 2:     **if** the target changes from π_{j-1} to π_j **then**
 3:         Label the out-of-date samples $\xi_1, \ldots \xi_m$ with corresponding weights $w_1, \ldots, w_m$.
 4:         **if** $j \notin D$ **then**
 5:             Update the weight $w_i \leftarrow w_i v_i$ where $v_i \propto \frac{d\pi_j}{d\pi_{j-1}}(\xi_i)$ for $i = 1, \ldots, m$.
 6:             Compute $d = \left(\sum_{k=1}^m w_k \right) / \left(\sum_{k=1}^m w_k^2 \right)$.
 7:             Set $w_i \leftarrow dw_i$ for $i = 1, \ldots, m$.
 8:             Write the weights into the sample database.
 9:         **if** $j \in D$ **then**
10:             Replace samples by $f_j(\xi_i, U_1), \ldots, f_j(\xi_m, U_m)$ where $U_1, \ldots, U_m \overset{iid}{\sim} U[0, 1]$, leaving the weights unchanged.
11:     **else** sleep for some time.

Suppose that the RMCMC process produces the samples $\xi_1, \ldots, \xi_m \sim \pi_{j-1}$ where π_{j-1} is the target of interest. Next, suppose the target changes from π_{j-1} to π_j. In order to use the samples from the previous measure, π_{j-1}, for estimating $\pi_j g_j$, the weights are updated as follows. For $i = 1, \ldots, m$ define the updated weight W_i from w_i as

$$W_i = w_i v_i \quad \text{where } v_i \propto \frac{d\pi_j}{d\pi_{j-1}}(\xi_i).$$

Afterwards, the weights are scaled such that the sum of the weights is equal to their effective sample size. More precisely, define the scaled weight $\widehat{w}_i$ from W_i as

$$\widehat{w}_i = W_i \frac{\sum_{k=1}^m W_k}{\sum_{k=1}^m W_k^2} \quad (i = 1, \ldots, m).$$

Table 2
Description of the global variables in the rolling MCMC system.

Variable	Description
RMCMC_ON	Indicator if RMCMC process is supposed to be producing new samples.
N_{MAX}	Maximum number of samples allowed in the database.
N	Number of samples contained in the database.
$(\xi_i, w_i)_{i=1,\dots,N}$	The samples and their corresponding weights in the database.

Algorithm 2.3 Control Process

    Parameters: $\beta_1, \beta_2, \gamma_1, \gamma_2, N_{\text{MIN}}$.
1: **repeat** indefinitely.
2:     Compute Q and A.
3:     **if** $A < \beta_1$ and $N \geq N_{\text{MIN}}$ **then** set RMCMC_ON=FALSE.
4:     **if** $(A > \beta_2)$ or (RMCMC_ON=FALSE and $Q < \gamma_1$ and $N = N_{\text{MIN}}$) **then** set RMCMC_ON=TRUE
5:     **if** RMCMC_ON=FALSE and $Q < \gamma_1$ and $N > N_{\text{MIN}}$ **then** decrease N_{MAX}.
6:     **if** RMCMC_ON=TRUE and $Q > \gamma_2$ **then** increase N_{MAX}.

Straightforward calculations show that scaling in this fashion ensures that the effective sample size of the database is the sum of the effective sample sizes of the most recently weighted samples and the newly generated samples.

2.5. Control process

The control process determines when the RMCMC process is paused and changes the maximum number of samples contained in the database. This is done to maintain the accuracy of the estimate of interest and the quality of the samples. We now discuss each of these in turn.

At any given time, denote the samples in the database by $\xi_1, \dots, \xi_N$. For $i = 1, \dots, N$ denote the ith sample weight in the database as w_i. To estimate the quantity of interest $\pi_k g_k$, for some $k \in \mathbb{N}$, we use the estimator

$$T = \frac{\sum\limits_{i=1}^{N} w_i g_k(\xi_i)}{\sum\limits_{i=1}^{N} w_i}.$$

The accuracy of the estimate, A, is defined as the standard deviation of T (in Section 2.7 we discuss how to estimate A). The process aims to control the accuracy A such that $A < \epsilon$ for some fixed $\epsilon > 0$. When considering multiple estimates i.e. multivariate g_k, we force the standard deviation of all the estimates below the threshold ϵ. One approach to control the accuracy would be to pause the RMCMC process each time $A < \epsilon$ and resume if $A \geq \epsilon$. However, this may lead to the RMCMC process being paused and resumed each time a new observation is revealed, as a small change in the accuracy will inevitably occur. Therefore, we use $0 < \beta_1 < \beta_2 \leq \epsilon$ so that if $A \leq \beta_1$ the RMCMC process is paused and if $A > \beta_2$ the RMCMC process is resumed.

The control process also controls the quality of the samples in the database. The process aims to hold a good mixture of samples in the hope that a future change of measure does not require the resuming of the RMCMC process. We define the quality of the samples in the database as

$$Q = \frac{\text{ESS}}{N_{\text{MAX}}} \quad \text{where ESS} = \frac{\left(\sum\limits_{i=1}^{N} w_i\right)^2}{\sum\limits_{i=1}^{N} w_i^2}.$$

The quality of the samples, Q, is the effective sample size of all the weights in the database divided by the optimal effective sample size of the database, N_{MAX}. The optimal effective size of the database consists of a database with N_{MAX} samples all with weight 1. As with the accuracy, we aim to maintain the quality such that $\gamma_1 < Q < \gamma_2$ for some $0 < \gamma_1 < \gamma_2 \leq 1$. The control process is summarised in Algorithm 2.3. To ensure that the database is never depleted, a minimum number of samples is imposed at $N_{\text{MIN}} > 0$ such that the number of samples, N and N_{MAX} cannot drop below N_{MIN}. Therefore, when the RMCMC process is paused, $Q < \gamma_1$ and $N_{\text{MAX}} = N_{\text{MIN}}$ we cannot decrease N_{MAX} any more. In this case, the RMCMC process is resumed to generate new samples that replace the poor quality samples in the database.

2.6. Deletion process

This process deletes samples from the database if the current number of samples, N, exceeds the maximum number of samples allowed N_{MAX}. Removing samples from the database reduces the computational work performed by the update

---

Algorithm 2.4 Deletion Process

1: **repeat** indefinitely.
2:     **if** $N > N_{MAX}$ **then** delete samples from the database.
3:     **else** sleep for some time.

---

process and calculating the estimates. Moreover, lowering the number of samples is the way the control process maintains the quality of the samples. For simplicity, if $N > N_{MAX}$, the $N - N_{MAX}$ samples that were produced the earliest are removed. The deletion process is summarised in Algorithm 2.4.

2.7. Modifying batch means to estimate the accuracy

There are several methods to estimate the variance of MCMCs such as block bootstrapping (Lahiri, 2003, Chapter 3), batch means (Flegal and Jones, 2010) and initial sequence estimators (Geyer, 1992). In our system the samples in the database have weights which complicates the estimation of the variance. The aforementioned methods cannot be used as they essentially treat all samples with equal weight. We now present a version of the batch mean approach that is modified to account for the sample weights.

Assume that the estimate of interest is $\pi_n g_n$ for some $n \in \mathbb{N}$. First, order the samples in the database $\xi_1, \ldots, \xi_N$ and their corresponding weights $w_1, \ldots, w_N$ by their production date. This ensures that the dependence structure of the samples is maintained. Then we divide the samples into batches or intervals of length b according to their weights. More precisely, let $D_0 = 0, D_j = \sum_{i=1}^{j} w_i$ and $L = \lceil \sum_{i=1}^{N} w_i / b \rceil$ be the number of batches. It may be the case that a weight spans more than one interval. Therefore we need to divide each weight by the proportion it spans a given interval. For the ith interval and uth sample define $\kappa_i(u) = [\min\{D_u, ib\} - \max\{D_{u-1}, (i-1)b\}]^+$, where $[x]^+ = \max(0, x)$, for $i = 1, \ldots, L$. Then $\kappa_i(u)$ is the batch weight of ξ_u in interval i. The mean of the weighted samples in the ith interval is

$$\widehat{\mu}_i = \frac{\sum_{u=1}^{N} \kappa_i(u) g_n(\xi_u)}{\sum_{u=1}^{N} \kappa_i(u)}.$$

Finally, we estimate the squared accuracy by

$$\hat{A}^2 = \frac{1}{L} \sum_{i=1}^{L} (\widehat{\mu}_i - \widehat{\mu})^2, \quad \widehat{\mu} = \frac{1}{L} \sum_{j=1}^{L} \widehat{\mu}_j.$$

The batch length b should be large enough to capture the correlation between samples, yet small enough to give a stable estimate of the variance. In practice we recommend using several batch lengths in order to get a conservative estimate of A. Moreover, the batch mean estimate should not be trusted when the number of batches, L, is low. This can occur as $\sum_{i=1}^{N} w_i$ can become very small. In this case, we suggest setting the accuracy A to -1 nominally. This prompts the control process to remove samples from the database and then restart the RMCMC process. This action effectively replenishes the database with new samples. In practice, we recommend taking this action when $L < 20$.

2.8. Remarks

Effective sample size for correlated samples. The quality, Q, uses the effective sample size defined for independent samples, not correlated samples which we use in our system. In the system, consider the extreme case where all samples have the same value i.e. $\xi_1 = \cdots = \xi_N$ produced from the same target. Each of these samples will have the same weight and therefore $Q = 1$ suggesting that the optimal quality has been achieved. Further, the accuracy of the estimate, A, will be very low since the weights and samples are all the same. Hence, in this extreme case, the control process would take no action. This is clearly undesirable. Ideally, the effective sample size used to calculate Q should take into account the autocorrelation of the samples, where high autocorrelation (in absolute value) leads to a lower effective sample size. However, we use this version of the effective sample size for independent samples as it is quick and simple to compute.

Degeneracy of the sample weights. We now discuss how the system handles two types of degeneracy of the sample weights. The first is where a single sample in the database has most of the total weight and all other samples have 0 or nearly 0 weight. If this were to occur, the effective sample size, and therefore the quality, Q, will be very low. In this case, the control process will remove samples from the database before resuming the RMCMC process. The second is where all sample weights are 0 or nearly 0. As a consequence, the sum of the weights, $\sum_{i=1}^{N} w_i$, will be very low. Recall that the batch mean approach uses $L = \lceil \sum_{i=1}^{N} w_i / b \rceil$ batches where b is the length of the batch. Further, if $L < 20$ the control process removes samples from the database and resumes the RMCMC process. Therefore, in the case where $\sum_{i=1}^{N} w_i$ drops to low, the sample database in replenished. To summarise, the system does not attempt to avoid these types of degeneracy, but to take remedial action when it does occur.

Burn-in periods. In the RMCMC process, we perform a burn-in each time a change of measure occurs. In some cases, however, it may not be necessary, as we now discuss. Assume that we have the samples $\xi_1, \ldots, \xi_m \sim \pi_{j-1}$. Next, consider a change of measure from π_{j-1} to π_j such that $j \in D$. In this case, a burn-in period is unnecessary as the new chain starts at a representative of π_j, namely $f_j(\xi_m, U) \sim \pi_j$ where $U \sim U[0, 1]$. On the other hand, if either the samples $\xi_1, \ldots, \xi_m$ are not from π_{j-1} or the transition function in operation $3'$, but not operation 3, is available, then a burn-in period is required. In any case, performing a burn-in is mostly harmless.

Subsampling. The samples produced from an MCMC method are correlated. If the correlation of the samples is high then a large number of samples are required to achieve the desired accuracy of the estimate. As a consequence, the update process and the calculation of the estimate would take a long time. To alleviate this problem we use subsampling.

Use of subsampling within an MCMC method entails saving only some samples produced. More precisely, with a sub-sampling size k, every kth sample is saved and the rest discarded. To choose the subsampling size k, we suggest performing a pre-initialisation run of the MCMC on the initial set of data. One approach, that we use in our implementation of the system, is to vary k until $\rho := \varsigma^2 / \text{var}\{g_1(\xi_1)\} \approx 2$ where $\varsigma^2 = \text{var}\{g_1(\xi_1)\} + 2\sum_{j=1}^{\infty} \text{cov}\{g_1(\xi_1), g_1(\xi_{1+j})\}$. We found that setting $\rho \approx 2$ worked well in our implementations of the system, however may not be appropriate in all applications. In practice, a method such as initial sequence methods (Geyer, 1992) or a batch mean approach (Brooks et al., 2011, Section 1.10.1) can be used to estimate ς^2. We chose to use the batch mean approach in our system.

If the initial Markov chain $\xi_1, \xi_2, \ldots$ is Harris recurrent and stationary with invariant distribution π, then by the Markov chain central limit theorem (e.g. Jones, 2004)

$$\sqrt{n}\left\{\frac{1}{n}\sum_{i=1}^{n} g(\xi_i) - \int g(x)\pi(dx)\right\} \xrightarrow{d} N\left(0, \varsigma^2\right) \quad \text{as } n \to \infty.$$

Thus ς^2 is the asymptotic variance of the Markov chain. Hence, by choosing $\rho \approx 2$, we obtain

$$2\sum_{j=1}^{\infty} \text{cov}\left\{g(\xi_1), g(\xi_{1+j})\right\} \approx \text{var}\left\{g(\xi_1)\right\},$$

i.e. the sum of all covariance terms contributes as much as $\text{var}\{g(\xi_1)\}$ to ς^2. This way, the covariance between the samples is prevented from getting too large relative to $\text{var}\{g(\xi_1)\}$.

Choice of Scaling. As discussed in Section 2.3, the database will consist of weighted samples from different target distributions. In Section 2.5 the weighted sample average, T, is used to estimate $\pi_j g_j$ for some $j \in \mathbb{N}$. In this subsection we show that, due to the scaling of the weights (Section 2.4), the variance of T is minimised under certain assumptions. A similar calculation can be found in Gramacy et al. (2010).

We begin by showing that T can be decomposed according to two sets of samples. Denote the invariant measure of the RMCMC process at a given time instance as π_j for some known $j \in \mathbb{N}$. Further, label the samples produced from this MCMC targeting π_j as $\xi_{m+1}, \ldots, \xi_N$ for some $m \in \{0, \ldots, N\}$. The case $m = N$ corresponds to the situation when no samples have been produced from π_j. Label the remaining sample as $\xi_1, \ldots, \xi_m$. These samples will have already been weighted and scaled in previous iterations.

The estimator, T, can be decomposed according to the two sets of samples as

$$T = \frac{\sum_{i=1}^{m} \widehat{w}_i g_j(\xi_i) + \sum_{i=m+1}^{N} g_j(\xi_i)}{\sum_{i=1}^{m} \widehat{w}_i + (N - m)},$$

as $w_j = 1$ for $j = m+1, \ldots, N$. In terms of the updated weights, T can be written as $T = \alpha T_1 + (1-\alpha)T_2$ where $T_1 = \sum_{i=1}^{m} W_i g_j(\xi_i) / \sum_{i=1}^{m} W_i$ and $T_2 = \sum_{i=m+1}^{N} g_j(\xi_i)/(N-m)$ are the individual estimators of $\pi_j g_j$ given by the two sets of samples and

$$\alpha = \frac{\text{ESS}_m}{\text{ESS}_m + (N - m)} \quad \text{where ESS}_m = \frac{\left(\sum_{i=1}^{m} W_i\right)^2}{\sum_{i=1}^{m} W_i^2}.$$

The choice of the scaling performed in the update process (Section 2.4) led to this choice of α. We now show that this choice of α, under certain assumptions, minimises the variance of T. Assume that $\phi \in \mathbb{R}$ is a constant. Then the variance of the estimator $T = \phi T_1 + (1-\phi)T_2$ is $\text{var}(T) = \phi^2\text{var}(T_1) + (1-\phi)^2\text{var}(T_2)$ where we assume that T_1 and T_2, or more specifically the two sets of samples $\xi_1, \ldots, \xi_m$ and $\xi_{m+1}, \ldots, \xi_N$, are independent. The variances of the individual estimators are

$$\text{var}(T_1) = \frac{\sigma^2}{\text{ESS}_m} \quad \text{and} \quad \text{var}(T_2) = \frac{\sigma^2}{N - m},$$

where we assume that $\text{var}\{g_j(\xi_i)\} = \sigma^2$, for $i = 1, \ldots, N$ and that the weights are constants. Upon differentiating we find that setting ϕ to $\text{ESS}_m/\{\text{ESS}_m + (N-m)\}$ minimises $\text{var}(T)$ thus regaining α. These assumptions are unrealistic in our setting.

However, this motivates the use of a burn-in period within the RMCMC process after new data are observed. Although we cannot guarantee independence between the sets of samples, the burn-in period at least weakens their dependence.

3. Application to a model of a football league

In this section we demonstrate how the system performs on a model of a football league. The data we use are the English Premier League results from 2005/06 to 2012/13 season. In a season, a team plays all other teams twice. For each match played, a team receives points based on the number of goals they and their opponent score. If a team scores more goals than their opponent they receive 3 points. If a team scores the same number of goals as their opponent they receive 1 point. If a team scores fewer goals than their opponent they receive 0 points. The rank of each team is determined by their total number of points, where the team with the highest number of points is ranked 1st. A tie of ranks is then determined by goal difference and then the number of goals scored.

We are interested in the probability of each rank position for all teams at the end of a season. The aim is to estimate these rank probabilities to a given accuracy. Thus, in this application we are concerned about maintaining the accuracy of multiple predictions.

Throughout this section, we use the following notation. Let I_p be the $p \times p$ identity matrix and 1_p be a vector of 1s of length p. Further, let $N(\mu, \Sigma)$ denote a multivariate normal distribution with mean μ and covariance matrix Σ. Denote the cardinality of a set A by $|A|$. We shall reserve the index $t = 1, \ldots, T$ for reference to seasons. Lastly, let $\log N(\mu, \sigma^2)$ denote a log-normal distribution i.e. if $X \sim N(\mu, \sigma^2)$ then $\exp(X) \sim \log N(\mu, \sigma^2)$.

We begin by presenting a model for football game outcomes. The model we use is similar to that presented in Glickman and Stern (1998) and Dixon and Coles (1997).

3.1. Football league model

Consider a model with hidden Markov process X_t ($t \in \mathbb{N}$), observed process Y_t ($t \in \mathbb{N}$) and parameter θ. The observation Y_t contains all observations for state X_t. Denote the jth observation of state t as $Y_{j,t}$. Next define the kth observation batch of state t as $\widetilde{Y}_{k,t}$ for $k = 1, \ldots, c_t$ for some $c_t \geq 1$. For instance, if the observations are batched in groups of 10, the kth batch of state t is $\widetilde{Y}_{k,t} = Y_{10k-9,t}, \ldots, Y_{10k,t}$. In this application section, we are interested in the model

$$\begin{cases} p(x_t|x_{1:t-1}, \theta) = p(x_t|x_{t-1}, \theta) \\ p(\widetilde{y}_{k,t}|\widetilde{y}_{1:(k-1),t}, y_{1:(t-1)}, x_{1:t}, \theta) = p(\widetilde{y}_{k,t}|x_t, \theta) \\ p(x_1|\theta), \ p(\theta) \end{cases} \tag{1}$$

where $\widetilde{y}_{1:0,t}$ is an empty observation batch introduced for notational convenience. In this section, the sequence of target distributions is defined as follows. Let $\varpi_{k,t} = p(x_{1:t}, \theta|\widetilde{y}_{1:k,t}, y_{1:(t-1)})$ for $t = 1, 2, \ldots$ and $k = 0, \ldots, c_t$. Then, we are interested in the targets $\pi_n = \varpi_{\varphi_1(n), \varphi_2(n)}$ for $n \in \mathbb{N}$ where

$$\varphi_2(n) = \max \left\{ j \in \mathbb{N} : (n-1) \geq \sum_{i=1}^{j-1} (c_i + 1) \right\}, \qquad \varphi_1(n) = n - 1 - \sum_{i=1}^{\varphi_2(n)-1} (c_i + 1),$$

where we set $\sum_{i=1}^{0} (c_i + 1) = 0$. The transition steps occur at $k \in D = \{n \in \mathbb{N} : \varphi_1(n) = 0\}$. In this application, the transition functions f_k ($k \in D$) are dictated by the model namely $p(x_t|x_{t-1}, \theta)$ in (1).

We now describe the states X_t, the observations Y_t and the parameter θ in this football application. Each team is assumed to have a strength value (in $\mathbb{R}$) which remains constant within a season. Let U_t be the set of teams that play in season t, $X_{i,t}$ be the strength of team i in season t and $X_t = (X_{i,t})_{i \in U_t}$. To condense notation, for any set $A \subset U_t$ define $X_{A,t} := (X_{i,t})_{i \in A}$ and form the parameter vector $\theta = (\lambda_H, \lambda_A, \sigma_p, \sigma_s, \eta, \mu_p)$, which we now define.

At the end of every season, some teams are relegated and new teams are promoted to the league. Denote the set of promoted teams that begin season t by W_t and let $V_t = U_t \backslash W_t$ be the set of teams that remain in the league from season $t - 1$ to t. The promoted teams' strengths are introduced such that $X_{W_t,t}|(\theta, X_{t-1} = x_{t-1}) \sim N\left(\mu_p 1_{|W_t|}, \sigma_p^2 I_{|W_t|}\right)$. Thus any previous history in the league is not used for a promoted team. From season $t - 1$ to t, the strengths of the teams that were not relegated are evolved such that $X_{V_t,t}|(\theta, X_{t-1} = x_{t-1}) \sim N\left(\eta C_t x_{V_t,t-1}, \sigma_s^2 I_{|V_t|}\right)$, where $C_t = I_{|V_t|} - |V_t|^{-1} 1_{|V_t|} 1_{|V_t|}^T$. Thus between seasons, the strengths of the teams that are not relegated are centred around 0 and expanded ($\eta > 1$) or contracted ($\eta < 1$). Next, consider a match, in season t, between home team j and away team k ($j, k \in U_t$). We assume that the number of home $G_{j,H}^k$ and away goals $G_{k,A}^j$ are modelled by $G_{j,H}^k|(\theta, X_t) \sim \text{Poisson}\left(\lambda_H \exp\left\{x_{j,t} - x_{k,t}\right\}\right)$ and $G_{k,A}^j|(\theta, X_t) \sim \text{Poisson}\left(\lambda_A \exp\left\{x_{k,t} - x_{j,t}\right\}\right)$ respectively independently of each other. The parameters λ_H and λ_A are strictly positive and pertain to the home and away advantage (or disadvantage) which is assumed to be the same across all teams and all seasons. More precisely, λ_H (λ_A) is the expected number of home (away) goals in a match between two teams of equal strength. Finally, denote the results of season t by Y_t; the number of home and away goals for all games in season t. For this football application, the sample space is $S_n = \mathbb{R}^{20\varphi_2(n)+2} \times (\mathbb{R}^+)^4$.

For the first season strengths, we use an improper flat prior. For the home and away advantage we take respective Gamma distribution priors of shapes 5 and 2 and scales 5 and 1. For (η, σ_s) and (μ_p, σ_p) we take their Jeffreys priors. Jeffreys prior was

used for both (η, σ_s) and (μ_p, σ_p) after considering the amount of information available for each parameter. For instance, if 10 seasons are considered, only 9 transitions between seasons are available for the likelihood of (η, σ_s). Thus, using an informative prior would greatly influence the posterior distribution. This can also be argued for the promotion parameters (μ_p, σ_p).

3.2. The MCMC step

For the MCMC step in the RMCMC process (Algorithm 2.1), we use a Metropolis–Hasting algorithm (Metropolis et al., 1953; Hastings, 1970). In general, a different, potentially more complex MCMC method can be used. However, the system does not rely on the choice of MCMC method, and will work with a simple sampler, as demonstrated in this application. We use independent proposal densities for the separate parameters. Due to the high dimension of the combine states and parameter, we choose to implement block updates (Brooks et al., 2011, Section 21.3.2). This entails proposing parts of the state and parameter at any stage. The proposal densities used and the block updating are summarised in Algorithm 2.1 located in Section 2 in the Supplementary material (see Appendix A). In the algorithm we propose a new strength of a single season 80% of the time and part of the parameter θ the remaining 20%. This was done so that exploration of the chain was mainly focused on the states. The proposal densities' parameters were determined by consideration of the acceptance rate in a pre-initialisation run of the MCMC. Lastly, the samples were written into the database in batches of 1000.

3.3. League predictions

In Section 3.1 we introduced a model for the team strengths and the outcome of football matches, in terms of goals scored. In Section 3.2 we presented the MCMC method which produces samples used to estimate the states and parameters of the model. We now explain how these samples are used to predict the vector of final ranks for the current season, which is our estimate of interest i.e. $\pi_n g_n$.

For each sample, all games in a season are simulated once. Thus each sample gives a predicted end of season rank table. The distribution across these predicted rank tables gives the estimated probabilities of the ranks of each team. This distribution is the posterior summary of interest whose accuracy we aim to control.

3.4. System parameters

As mentioned in Section 2.8, we performed a pre-initialisation run using 10,000 samples to determine the subsampling size. Based on the results from the 2005/06 to the 2009/10 season, we found that a subsample size of 80 gave $\varsigma \approx 2$. We used a burn-in period of $B_0 = 10,000$ within the RMCMC process. Within the control process we use $\beta_1 = 0.01$ and $\beta_2 = 0.0125$ for the accuracy thresholds and $\gamma_1 = 0.1$ and $\gamma_2 = 0.75$ for the quality thresholds. Whenever the control process demanded a change in N_{MAX}, it was increased or decreased by 10% of its current value. Finally, we set $N_{\text{MIN}} = 1000$.

As mentioned in Section 3.3, our estimate consists of rank probabilities for each team i.e. each team has estimated probabilities for ending the season ranked 1st, ..., 20th. The accuracy of each of the 400 rank probabilities is calculated using the method presented in Section 2.7 using two batch lengths $b = 10$ and $b = 50$. The maximum standard deviation is reported as the accuracy of the estimate to be conservative.

3.5. Results

The system is initialised with the results from the 2005/06 to 2009/10 seasons of the English Premier League. Using the samples from this initialisation, we proceeded with 3 separate runs of the system. The system itself remained unchanged in each of the runs; however, the way the results for the next 2 seasons were revealed varied. The match results were revealed individually, in batches of 7 days and in batches of 30 days. A new data batch was revealed only if the RMCMC process was paused.

In Table 3 we present the system results of each run. We see that for larger data batches, the RMCMC process is resumed more often. Further, the percentage of new samples generated after new data are revealed increases as with the size of the data batch. The average percentage of new samples is calculated as follows. Before a new data batch is revealed the percentage of new samples in the database generated after the introduction of the latest data is calculated. The average of these percentages is then taken over the data batches. This means that for larger data batches the RMCMC process will often be resumed to generate new samples that replace most of the samples already in the database. In Table 4 we present the estimated posterior mean of the components of θ at the end of the run for each batch size. As expected, being based on the same data, these final estimates are almost identical for the various batch sizes. In Table 5 we present the predicted end of 2012/13 season ranks for selected teams and ranks. Each team and rank have 3 predictions given by the runs using different batch sizes. For each batch size, these predictions are being controlled. More precisely, for every rank of every team the predictions' standard deviation is being controlled below $\beta_2 = 0.0125$. This is consistent with the predictions across the various batch sizes. The predictions for all teams and ranks can be found in Section 1 in the Supplementary material (see Appendix A). In the following, we present some results for the 7 day batch run only. Further results for all the batch sizes are presented in Section 1 in the Supplementary material (see Appendix A). In Fig. 1 we display the accuracy of the predictions (A), the quality of the samples (Q) and the number of samples in the database (N) as new data are revealed. In Fig. 1(a), the

Table 3
System summary for various data batch sizes.

	Individual	7 day	30 day
No. of batches	760	70	20
Range of games per batch	[1, 1]	[3, 21]	[10, 53]
No. times RMCMC resumed	39	31	18
Total No. MCMC steps	24,010,000	14,230,000	9,240,000
Average % of new samples	2%	20%	53.6%

Table 4
End of run parameter estimated mean with 95% credible intervals.

Parameter	Individual	7 day	30 day
λ_H	1.446 (1.406, 1.497)	1.447 (1.406, 1.496)	1.446 (1.406, 1.494)
λ_A	1.031 (0.998, 1.073)	1.032 (0.995, 1.073)	1.032 (0.997, 1.077)
η	0.970 (0.865, 1.054)	0.967 (0.865, 1.049)	0.964 (0.864, 1.048)
σ_s	0.083 (0.061, 0.117)	0.084 (0.059, 0.113)	0.086 (0.059, 0.116)
$-\mu_p$	0.245 (0.316, 0.172)	0.242 (0.322, 0.167)	0.244 (0.315, 0.171)
σ_p	0.116 (0.049, 0.204)	0.117 (0.063, 0.191)	0.114 (0.06, 0.202)

Table 5
End of 2012/13 season rank predictions for selected teams and ranks. Each team and rank have 3 predictions given by (from top to bottom) the individual, 7 day and 30 day batch run.

Team	Rank					
	1	2	3	$\cdots$	18	19
Arsenal	8%	14%	17%	$\cdots$	0%	0%
	8%	14%	17%	$\cdots$	0%	0%
	8%	15%	19%	$\cdots$	0%	0%
Aston Villa	0%	0%	1%	$\cdots$	6%	5%
	0%	0%	1%	$\cdots$	6%	5%
	0%	0%	1%	$\cdots$	6%	5%
Chelsea	9%	15%	19%	$\cdots$	0%	0%
	9%	15%	21%	$\cdots$	0%	0%
	10%	16%	20%	$\cdots$	0%	0%
Everton	1%	2%	6%	$\cdots$	1%	1%
	1%	3%	5%	$\cdots$	1%	1%
	1%	2%	5%	$\cdots$	1%	1%
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Wigan	0%	0%	0%	$\cdots$	10%	11%
	0%	0%	0%	$\cdots$	10%	11%
	0%	0%	0%	$\cdots$	11%	12%

control process attempts to keep the accuracy of the predictions between $\beta_1 = 0.01$ and $\beta_2 = 0.0125$. Occasionally, after new data are revealed, the accuracy exceeds the upper threshold β_1. The accuracy drops nominally to 0 at the end of each season prior to the introduction of the next seasons' fixtures. Similarly, in Fig. 1(b), the quality of the samples is attempted to be kept between $\gamma_1 = 0.1$ and $\gamma_2 = 0.75$. In Fig. 1(c), we see that the number of samples in the database, N, varies over time. More precisely, after 5 batches of data, 19,246 samples are used. However, later the number of samples used decreases to approximately 14,000 samples. Similar features are seen for the different batch sizes. The change in the accuracy of the predictions and the quality of the samples gets smaller as the batch size decreases.

Fig. 1(d) is a plot of the Kaplan–Meier estimator (Kaplan and Meier, 1958) of the survival function of the samples in the database as new data are revealed. More precisely, let U be a random variable of the number of new data batches observed before a sample is deleted. Then Fig. 1(d) is a plot of the Kaplan–Meier estimator of $S(u) = P(U > u)$. The Kaplan–Meier estimator takes into consideration the right-censoring due to the end of the simulation i.e. samples that could have survived longer after the simulation ended. We see that samples survive as new data are observed e.g. a sample survives 10 or more batches with probability 0.33. Thus samples are reused as envisaged in Section 2.4. Lastly, from using the different batch sizes (see Section 1 in the Supplementary material, Appendix A) we see that samples survive more data batches as the size of the batch gets smaller.

In order to determine the quality of the predicted ranks given by the system, we performed a separate run and consider the coverages of the prediction intervals. For this run the initial observations consisted of the 2005/06–2007/08 seasons' results. We then introduced the match results for the 2008/09–2012/13 seasons in 7 day batches. Over the 5 seasons there
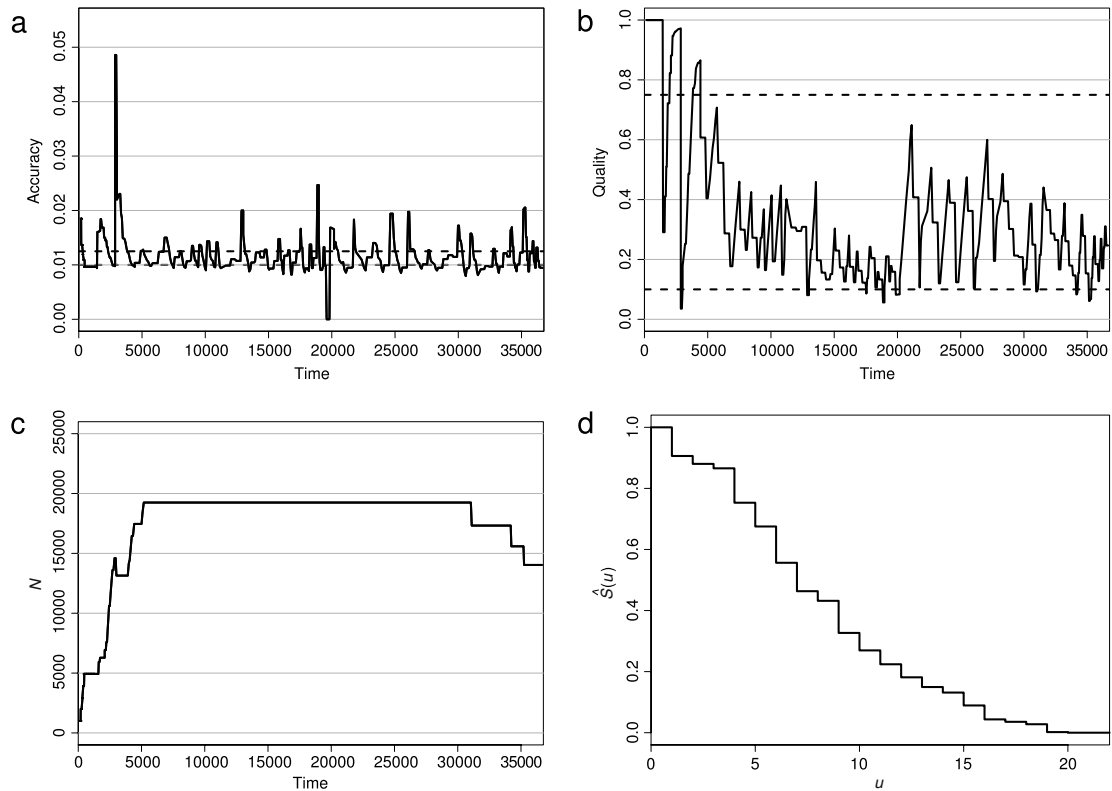
Fig. 1. System results using 7 day batches where (a) is the accuracy (A) of the predictions, (b) is the quality (Q) of the samples, (c) is the number of samples in the database (N) as new data are revealed and (d) is the Kaplan–Meier estimator the samples lifetime.

were 178 batches of intervals for each team. Before each batch of results was introduced, conservative 50% and 95% intervals were formed for the predicted end of season rank of each team. These confidence intervals are conservative due to the discreteness of ranks. The true mass contained in the conservative 50% intervals was on average 76.1%. Similarly, the true mass contained in the conservative 95% intervals was on average 98.8%. When compared with the true end of season ranks, 74.2% of the true ranks lay in the conservative 50% intervals and 99.3% lay in the 95% intervals.

4. Application to a linear Gaussian model

In Section 3 we are unable to check if the strengths of the teams and the other parameters (i.e. the states and parameters) are being estimated accurately as their true distributions are unknown. In this section we inspect the estimates given by the RMCMC system using simulated data. We use a linear Gaussian model such that the Kalman filter (Kalman, 1960) can be applied. This simulation will allow us to compare the RMCMC system and the Kalman filter estimates. This linear and Gaussian model was chosen to resemble the football model described in Section 3. For this model the Kalman filter gives the exact conditional distribution. Therefore, the Kalman filter will provide the benchmark estimates to compare against.

Consider the model defined as follows:

$$\begin{cases} \text{State} : X_t = AX_{t-1} + \Phi_t, & \Phi_t \overset{\text{iid}}{\sim} N(0, \Sigma) \\ \text{Observation} : Y_t = BX_t + \Psi_t, & \Psi_t \overset{\text{iid}}{\sim} N(0, \Xi), \end{cases} \quad \text{for } t = 1, 2, \dots \tag{2}$$

and prior distribution $X_0 \sim N(\mu_0, \Sigma_0)$. For this particular simulation we chose $A = 0.7(\mathbb{I}_{20} - \frac{1}{20} 1_{20} 1_{20}^T)$, $\Sigma = 0.05 \mathbb{I}_{20}$ and $\Xi = 0.02 \mathbb{I}_{380}$. The matrix B is constructed according to the football matches in the English Premier League in the 2005/06 season. More precisely, each row of B consists of zeros apart from two entries at i and j corresponding to a football match between home team i and away team j. A 2 is put in the ith position and a 1 at the jth. The rows are ordered chronologically from top to bottom. For the prior distribution, we set μ_0 to be a vector of zeros and $\Sigma_0 = \mathbb{I}_{20}$. Denote the ith component of X_t as $X_{i,t}$.

A single realisation of the states and observations was generated for $t = 1, \dots, 7$. Using these observations, the RMCMC system was run 100 times to estimate means. This was compared with the estimates given by the Kalman filter. Each run of the RMCMC system was initialised using the observations from state $t = 1, \dots, 5$.
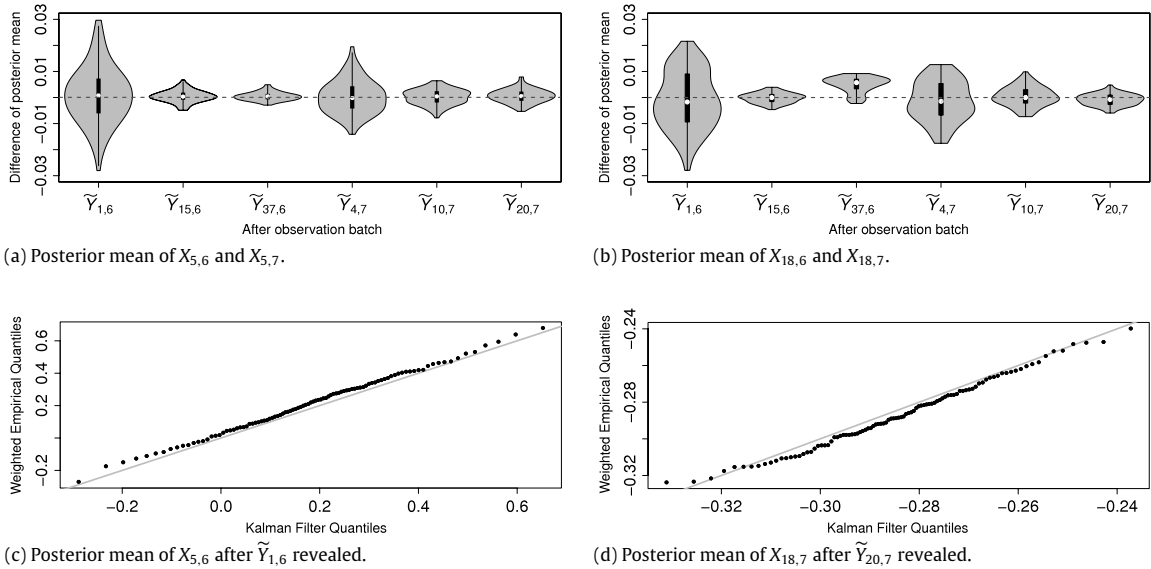
(a) Posterior mean of $X_{5,6}$ and $X_{5,7}$.

(b) Posterior mean of $X_{18,6}$ and $X_{18,7}$.

(c) Posterior mean of $X_{5,6}$ after $\widetilde{Y}_{1,6}$ revealed.

(d) Posterior mean of $X_{18,7}$ after $\widetilde{Y}_{20,7}$ revealed.

Fig. 2. Simulation results: (a) and (b) are violin plots of the difference between the 100 RMCMC and the Kalman filter posterior mean. (c) and (d) are Q–Q plots of the Kalman filter and RMCMC system posterior distribution from a single run (black).

The sequence of targets is similar to that used in Section 3.1 with $\varpi_{k,t} = p(x_{1:t}|\widetilde{y}_{1:k,t}, y_{1:(t-1)})$. For the transition function f_j ($j \in D$) we use the observation equation in (2). Finally, we take g_n to be the identity function, so that our estimate of interest in the posterior mean. The observations were revealed in batches of 10, so that each state consisted of 38 batches. Specifically, the vector Y_t contains all 380 observations where we denote the jth observation as $Y_{j,t}$. The kth observation batch of state t is $\widetilde{Y}_{k,t} := Y_{10k-9,t}, \ldots, Y_{10k,t}$. Therefore, after initialisation, the batches $\widetilde{Y}_{1,6}, \ldots, \widetilde{Y}_{38,6}, \widetilde{Y}_{1,7}, \ldots \widetilde{Y}_{38,7}$ are revealed.

Within the control process we again use $\beta_1 = 0.01$, $\beta_2 = 0.0125$ and $\gamma_1 = 0.1$, $\gamma_2 = 0.75$. Also, we set $N_{\text{MIN}} = 1000$. For this simulation controlling the accuracy A pertains to controlling the mean posterior of each component of every state as new data are observed. We use a Gibbs sampler (see e.g. Geman and Geman, 1984) as the conditional distributions for the states can be explicitly computed for this model. Each Gibbs sampler step consists of updating a single randomly chosen state as outlined in Algorithm 2.2 located in Section 2 of the Supplementary material (see Appendix A). We used no subsampling and used a burn-in period of $B_0 = 1000$. The accuracy was calculated using the batch mean approach described in Section 2.7 with batch lengths 10 and 25. The RMCMC process wrote 500 samples to the database at a time.

Fig. 2 presents results comparing the Kalman filter estimates with the 100 RMCMC estimates as the observations are revealed. The upper row of Fig. 2 are violin plots (see e.g. Hintze and Nelson, 1998) of the difference between the Kalman filter and the 100 RMCMC system posterior mean of selected states and components. Violin plots are smoothed histograms on either side of a box plot of the data.

The estimate may be biased due to the scaling and normalisation of the weights carried out by the update process (Section 2.4) (see for example Hesterberg, 1995 for the bias in weighted importance sampling). This is apparent in posterior mean for $X_{18,6}$ (Fig. 2(a)), as in 81 out of the 100 runs the RMCMC process remained paused after $\widetilde{Y}_{37,6}$ was revealed. For these 81 runs, the posterior mean was formed using weighted importance sampling. In contrast, we see nearly no bias in the posterior mean for $X_{5,6}$ after $\widetilde{Y}_{1,6}$ was revealed (Fig. 2(b)) where the RMCMC process was started in every run (the posterior mean given by the Gibbs sampler is unbiased). Table 6 shows the estimated bias of the 100 RMCMC system posterior means with respect to the estimate given by the Kalman filter. Table 7 shows the standard deviation of the 100 RMCMC system posterior means. We see that the standard deviation (the accuracy A) is controlled below the imposed threshold of $\beta_2 = 0.0125$. The lower row of Fig. 2 is Q–Q plots of the Kalman filter estimate and the weighted RMCMC samples' posterior distribution at the 1%, 2%, . . . , 99% quantile from 1 of the 100 runs. The Q–Q plots for other components of and RMCMC runs are similar to those presented. These Q–Q plots indicate that the two distributions are roughly similar.

Comparison of the two distributions is difficult as the RMCMC samples are not only weighted but are also dependent. Thus tests, such as the Kolmogorov–Smirnov test (e.g. see p. 35 Lehmann and D'Abrera, 1975), cannot be applied.

5. Conclusion

We have presented a new method that produces estimates from a sequence of distributions that maintains the accuracy at a user-specified level. In Section 3 we demonstrated that the system is not resumed each time an observation is revealed and thus the samples are reused. Therefore, we proceed with importance sampling whenever possible. Further we attempt to reduce the size of the sample database whenever possible (Section 2.5), thus limiting the computational effort of the update process and calculation of the estimates or predictions. In Section 4 we used a linear Gaussian model to show that the system

Table 6
Tables of estimated bias of the 100 system posterior means with respect to the Kalman filter posterior mean.

	Last batch revealed				Last batch revealed		
	$\widetilde{Y}_{1,6}$	$\widetilde{Y}_{15,6}$	$\widetilde{Y}_{37,6}$		$\widetilde{Y}_{1,6}$	$\widetilde{Y}_{15,6}$	$\widetilde{Y}_{37,6}$
$X_{5,6}$	0.0009	0.0004	0.0004	$X_{5,6}$	0.0009	0.0004	0.0004
$X_{18,6}$	−0.0012	−0.0001	0.0049	$X_{18,6}$	−0.0012	−0.0001	0.0049

Table 7
Tables of the empirical standard deviation of posterior mean given by 100 runs of the system.

	Last batch revealed				Last batch revealed		
	$\widetilde{Y}_{1,6}$	$\widetilde{Y}_{15,6}$	$\widetilde{Y}_{37,6}$		$\widetilde{Y}_{3,7}$	$\widetilde{Y}_{10,7}$	$\widetilde{Y}_{20,7}$
$X_{5,6}$	0.0109	0.0023	0.0016	$X_{5,7}$	0.0065	0.0030	0.0026
$X_{18,6}$	0.0110	0.0019	0.0030	$X_{18,7}$	0.0077	0.0037	0.0022

produced comparable estimates to those given by the Kalman filter. Proving the exactness of the estimates produced by the system, if possible, is a topic for future work.

For our system, we advocate using a standard MCMC method such as a Metropolis–Hastings algorithm before resorting to another more complicated method such as particle MCMC methods (Andrieu et al., 2010) or SMC[2] (Chopin et al., 2013). By starting with a standard MCMC approach, we avoid choosing the number of particles, choosing the transition densities and the resampling step that comes with using a particle filter, not to mention the higher computational cost.

Acknowledgements

Appendix A. Supplementary material

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.csda.2014.06.010.

References

Andrieu, C., Doucet, A., Holenstein, R., 2010. Particle Markov chain Monte Carlo methods. J. R. Stat. Soc. Ser. B 72 (3), 269–342.
Brooks, S., Gelman, A., Jones, G., Meng, X., 2011. Handbook of Markov Chain Monte Carlo. CRC Press.
Chopin, N., 2002. A sequential particle filter method for static models. Biometrika 89 (3), 539–552.
Chopin, N., Jacob, P.E., Papaspiliopoulos, O., 2013. SMC2: an efficient algorithm for sequential analysis of state space models. J. R. Stat. Soc. Ser. B 75 (3), 397–426.
Del Moral, P., Doucet, A., Jasra, A., 2006. Sequential Monte Carlo samplers. J. R. Stat. Soc. Ser. B 68 (3), 411–436.
Dixon, M.J., Coles, S.G., 1997. Modelling association football scores and inefficiencies in the football betting market. J. R. Stat. Soc. Ser. C 46 (2), 265–280.
Doucet, A., de Freitas, N., Gordon, N. (Eds.), 2001. Sequential Monte Carlo Methods in Practice. Springer.
Flegal, J.M., Jones, G.L., 2010. Batch means and spectral variance estimators in Markov chain Monte Carlo. Ann. Statist. 38 (2), 1034–1070.
Geman, S., Geman, D., 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. (6), 721–741.
Geyer, C.J., 1992. Practical Markov chain Monte Carlo. Statist. Sci. 7 (4), 473–483.
Glickman, M.E., Stern, H.S., 1998. A state-space model for National Football League scores. J. Amer. Statist. Assoc. 93 (441), 25–35.
Gordon, N., Salmond, D., Smith, A., 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEEE Proc. F Radar Signal Process. 140 (2), 107–113.
Gramacy, R., Samworth, R., King, R., 2010. Importance tempering. Stat. Comput. 20 (1), 1–7.
Hastings, W.K., 1970. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57 (1), 97–109.
Hesterberg, T., 1995. Weighted average importance sampling and defensive mixture distributions. Technometrics 37 (2), 185–194.
Hintze, J.L., Nelson, R.D., 1998. Violin plots: a box plot-density trace synergism. Amer. Statist. 52 (2), 181–184.
Jones, G.L., 2004. On the Markov chain central limit theorem. Probab. Surv. 1, 299–320.
Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. J. Basic Eng. 82 (1), 35–45.
Kaplan, E., Meier, P., 1958. Nonparametric estimation from incomplete observations. J. Amer. Statist. Assoc. 53, 457–481.
Kitagawa, G., 2014. Computational aspects of sequential Monte Carlo filter and smoother. Ann. Inst. Statist. Math. 1–29.
Lahiri, S., 2003. Resampling Methods for Dependent Data. Springer.
Lehmann, E.L., D'Abrera, H., 1975. Nonparametrics: Statistical Methods Based on Ranks. Holden-Day.
Liu, J.S., 2008. Monte Carlo Strategies in Scientific Computing. Springer.
Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equation of state calculations by fast computing machines. J. Chem. Phys. 21 (6), 1087–1092.