

NPL REPORT MS 21

**USER MANUAL FOR MCMCMH: SOFTWARE IMPLEMENTING A
METROPOLIS-HASTINGS ALGORITHM**

K JAGAN AND A B FORBES

APRIL 2018

User manual for MCMCMH: Software implementing a Metropolis-Hastings algorithm

K Jagan and A B Forbes
Data Science Group

April 2018

ABSTRACT

This report constitutes a *user manual* for software developed at the National Physical Laboratory to generate a sample from a user defined target distribution using the Metropolis-Hastings Markov chain Monte Carlo algorithm.

NPL Report MS 21

© NPL Management Limited, 2018

ISSN 1754–2960

National Physical Laboratory,
Hampton Road, Teddington, Middlesex, United Kingdom TW11 0LW

Extracts from this report may be reproduced provided the source is acknowledged and the
extract is not taken out of context

We gratefully acknowledge the financial support of the UK Department for Business,
Energy & Industrial Strategy

Approved on behalf of NPLML by Louise Wright,
Science Area Leader for Modelling

Contents

1	Introduction	1
1.1	Scope	1
1.2	Software user licence agreement	1
2	The Metropolis-Hastings algorithm	1
3	Software implementation	4
3.1	Generating the MCMC sample	5
3.1.1	mcmcmh	5
3.1.2	mcmcmhic	6
3.2	Convergence indices	7
3.3	Sample summary	8
3.4	Jumping distributions	9
3.4.1	Gaussian random walk	9
3.4.2	Gaussian independence chain	10
3.5	Target distributions	10
3.5.1	tar.m	11
4	Numerical example	12

1 Introduction

1.1 Scope

This document describes MATLAB software implementing the Metropolis-Hastings Markov chain Monte Carlo (MCMC) algorithm to generate samples from a user defined target distribution. The software generates multiple chains of samples and assesses convergence to the target distribution.

Section 2 describes the Metropolis-Hastings algorithm, Section 3 describes the individual software components and numerical examples are given in Section 4.

1.2 Software user licence agreement

The software is provided with a software user licence agreement and the use of the software is subject to the terms laid out in that agreement. By running the software, the user accepts the terms of the agreement.

2 The Metropolis-Hastings algorithm

Bayes' theorem states that the posterior distribution $p(\mathbf{a}|\mathbf{y})$ for parameters \mathbf{a} given data \mathbf{y} is

$$p(\mathbf{a}|\mathbf{y}) = K^{-1}p(\mathbf{y}|\mathbf{a})p(\mathbf{a}), \quad K = \int p(\mathbf{y}|\mathbf{a})p(\mathbf{a}) \, d\mathbf{a}.$$

For all but simple problems, the key difficulty in working with the posterior distribution is in determining the normalising constant K . This motivates the need for Markov chain Monte Carlo (MCMC) simulation methods which generate $\{\mathbf{a}_q\}$ samples from the posterior $p(\mathbf{a}|\mathbf{y})$ or indeed any distribution $p(\mathbf{a})$. In particular, the Metropolis-Hastings MCMC algorithm can be used to generate samples from distributions that are only known up to a normalising constant.

MCMC methods are iterative schemes that produce samples in such a way that values at a particular iteration \mathbf{a}_{q+1} only depend on the values at the previous iteration \mathbf{a}_q , i.e., they satisfy the Markov property. The chains must be designed in such a way that they converge to the target distribution $p(\mathbf{a})$. One commonly used method to achieve this is the Metropolis-Hastings MCMC method.

Suppose we wish to sample \mathbf{a}_q from a *target distribution* $p(\mathbf{a})$. Given a sample of \mathbf{a}_{q-1} , a proposed new sample \mathbf{a}^* is drawn at random from a *jumping distribution* $p_0(\mathbf{a}|\mathbf{a}_{q-1})$. Then \mathbf{a}_q is set to \mathbf{a}^* with acceptance probability

$$P_q = \min\{1, r_q\}, \quad r_q = \frac{p(\mathbf{a}^*)p_0(\mathbf{a}_{q-1}|\mathbf{a}^*)}{p(\mathbf{a}_{q-1})p_0(\mathbf{a}^*|\mathbf{a}_{q-1})}. \quad (1)$$

The simplest way to implement the acceptance step is to draw u_q from the uniform distribution $R(0, 1)$ and if $u_q < r_q$, set $\mathbf{a}_q = \mathbf{a}^*$, otherwise set $\mathbf{a}_q = \mathbf{a}_{q-1}$. The role of the acceptance probability is to ensure that when the direction of time is reversed, the behaviour of the process remains the same. This *reversibility* property leads to $p(\mathbf{a})$ being the limiting distribution of the chain.

The important practical feature of this acceptance probability is that $p(\mathbf{a})$ and $p_0(\mathbf{a}^*|\mathbf{a})$ need only be known up to a constant since $p(\mathbf{a})$ appears in the ratio $p(\mathbf{a}^*)/p(\mathbf{a}_{q-1})$, etc. The Metropolis-Hastings algorithm is also useful for sampling from complex target distributions that cannot be sampled from directly, using a jumping distribution from which it is easier to sample. The percentage of samples that are accepted is larger if the jumping distribution is closer to the target distribution.

We consider two types of jumping distributions - random walk and independence chain. If the jumping distribution depends on the sample at the previous iteration \mathbf{a}_{q-1} , this is said to be a random walk algorithm; e.g., if the proposed sample is generated from a Gaussian distribution centered around the sample from the previous iteration. If samples are generated independently of the previous iteration this is said to be an independence chain algorithm; e.g., if the proposed sample is generated from a Gaussian distribution centered around some fixed value for every iteration. Since the jumping distribution in this case is independent of the previous iteration, the Metropolis-Hastings acceptance ratio in Equation (1) reduces to

$$r_q = \frac{p(\mathbf{a}^*)p_0(\mathbf{a}_{q-1})}{p(\mathbf{a}_{q-1})p_0(\mathbf{a}^*)}. \quad (2)$$

After a number of iterations that allow the Markov chain to converge, the sampled $\{\mathbf{a}_q\}$ are considered to be sampled from the target distribution. This initial number of iterations is known as the “burn-in” period. A burn-in period of length M_0 implies that, for $q > M_0$, \mathbf{a}_q is considered to be a sample from the target distribution.

Test on convergence. The following scheme can be used to check the convergence of a chain by comparing the behaviour of chains of the same length generated using different starting points [1, section 11.6]. Suppose that we have samples $\mathbf{a}_{q,r}$, $q = 1, 2, \dots, M$, $r = 1, \dots, N$, from N chains of length M .

For each parameter $a = a_j$, we make the following calculations:

$$\bar{a}_{.,r} = \frac{1}{M} \sum_{q=1}^M a_{q,r}, \quad \bar{a}_{..} = \frac{1}{N} \sum_{r=1}^N \bar{a}_{.,r}, \quad B = \frac{M}{N-1} \sum_{r=1}^N (\bar{a}_{.,r} - \bar{a}_{..})^2,$$

and

$$s_r^2 = \frac{1}{M-1} \sum_{q=1}^M (a_{q,r} - \bar{a}_{.,r})^2, \quad W = \frac{1}{N} \sum_{r=1}^N s_r^2.$$

The quantity B represents the variance between the chains, and W the variance within the chains. The variance of the distribution associated with $a|y$ is estimated by

$$V^+ = \frac{N-1}{N}W + \frac{1}{N}B.$$

If the variance for the proposal $\hat{p}(a)$ distribution is greater than the target distribution (as recommended to ensure that the whole of $p(a)$ is sampled), then this estimate will represent an overestimate, but is unbiased in the limit as $M \rightarrow \infty$. On the other hand, the within variance $V^- = W$ can be expected to represent an underestimate because, for finite M , each chain will not have had an opportunity to range over all the target distribution. As $M \rightarrow \infty$, we expect the ratio

$$R = \left(\frac{V^+}{V^-} \right)^{1/2},$$

to approach 1 from above. This ratio represents the potential reduction in the estimate of the standard deviation of the distribution for $a|y$ as $M \rightarrow \infty$. If R is less than 1.05, the expected improvement in the estimate of the standard deviation by letting the chains run longer will be no more than 5 %.

A note on the various jumping distributions The closer the jumping distribution is to the target distribution, the more efficient is the algorithm, i.e. the higher the acceptance probability of proposed samples and the faster the convergence. Some considerations need to be made before choosing a jumping distribution.

- Support of the parameters - for instance a jumping distribution for a variance parameter should sample positive values.
- Symmetry of the distribution - if the target distribution is likely to be skewed it may be more efficient to use a skewed jumping distribution.
- Random walk or independence chain - a random walk algorithm generates a sample that is dependent on the sample at the previous iteration of the MCMC algorithm. In the case of the independence chain algorithm, samples generated are independent of previous samples. The random walk scheme explores the parameter space more efficiently and is less likely to get stuck at one point. Thus for a poor jumping distribution, the Metropolis-Hastings algorithm is likely to accept more samples using a random walk scheme than an independence chain scheme. To improve the chances of the independence chain scheme sampling from the tails of a distribution an over-dispersed jumping distribution is recommended. If the jumping distribution is a good approximation to the target distribution then an independence chain scheme could result in faster convergence than a random walk scheme.

3 Software implementation

The Metropolis-Hastings (MH) algorithm is implemented through several MATLAB modules. The primary modules `mcmcmh.m` and `mcmcmhic.m` generate Markov chains associated with the target distribution. The former generates samples using a jumping distribution which could either be based on a random walk algorithm or an independence chain algorithm. The latter implements an independence chain algorithm. In the case of independence chains, since the proposed sample at the current iteration does not depend on the sample from the previous iteration, the MH acceptance ratio is given by Equation (2). Hence, at each iteration, we only need to compute the jumping density at the current sample $p_0(\mathbf{a}^*)$ as the density evaluated at the previous stage will already be stored as $p_0(\mathbf{a}_{q-1})$. This approach reduces the computation time. Hence, if the user would like to run an independence chain algorithm, they are advised to use the `mcmcmhic.m` routine. `mcmcmh.m` runs a general MH algorithm; the jumping distribution determines whether an independence chain or random walk algorithm is used.

The other functions called by `mcmcmh.m` and `mcmcmhic.m` are:

- `mcmcci.m`: evaluates convergence indices
- `mcsums.m`: calculates summary statistics

The MCMC algorithms require the user to provide function handles for the target and jumping distributions. Two example jumping distributions have been provided:

- `jumpwrg.m`: implements a Gaussian random walk jumping distribution to be used as a function handle in `mcmcmh.m`
- `jumpicg.m`: implements a Gaussian independence chain jumping distribution to be used as a function handle in `mcmcmhic.m`

An example target distribution has also been provided:

- `tar.m`: the logarithm of the posterior distribution for $\log(\alpha)$ and $\log(\delta)$ for the model $\mathbf{y} \sim N(\alpha\delta, \sigma_0^2)$ assuming Gamma priors for α and δ .

Test scripts that perform MCMC sampling from the target distribution `tar.m` using the random walk and independence chain algorithms have also been provided in `r_example_rw.m` and `r_example_ic.m` respectively.

Details of the modules are provided in the sections that follow.

3.1 Generating the MCMC sample

3.1.1 `mcmcmh`

The software component `mcmcmh.m` has calling syntax

```
[S, aP, Rh, Ne, AA, IAA] = mcmcmh(target, jump, M, N, M0, Q, A0) .
```

This component generates N MCMC chains of length M . It generates proposal samples \mathbf{a}^* according to the jumping distribution, and evaluates their acceptance probabilities defined in Equation (1). An accept-reject procedure is carried out to determine whether or not \mathbf{a}^* is selected to be part of the MCMC sample. The outputs of the jumping distribution are proposed samples and the log of the ratio $p_0(\mathbf{a}_{q-1}|\mathbf{a}^*)/p_0(\mathbf{a}^*|\mathbf{a}_{q-1})$.

Once the sample has been generated, they are passed to `mcmcci.m` which determines convergence indices and to `mcsums.m` which evaluates summary statistics. Details of these modules are provided in the following subsections.

	Size	Description
		Inputs
target	function handle	This function takes as input array $A(n, N)$ and returns a vector $\log p(1, N)$ where $\log p(j)$ is the logarithm of target distribution evaluated at $A(:, j)$, up to an additive constant.
jump	function handle	This function takes as input array $A(n, N)$, samples at the current iteration, and returns $A^*(n, N)$, proposed samples, and a $1 \times N$ vector $dp0$. $dp0(j)$ is the difference between the logarithm of the jumping distribution associated with moving from $a = A(:, j)$ to $a^* = A^*(:, j)$ and that associated with moving from a^* to a , up to an additive constant. $\log(p_0(a a^*)) - \log(p_0(a^* a))$
M		Length of the chains.
N		Number of chains.
M0		Integer M_0 specifying the ‘burn-in’ period. Constraint: $M > M_0 \geq 0$.
Q	$nQ \times 1$	Quantiles to be evaluated. Ranges from 0 (minimum of sample) to 100 (maximum of sample) and 50 represents the median value
A0	$n \times N$	Array of feasible starting points: the target distribution evaluated at $A0(:, j)$ is strictly positive.
		Output
S	$(2 + nQ) \times n$	Summary statistics pertaining to the samples from the posterior distribution: mean, standard deviation and percentile limits, where the percentile limits are given by Q.
aP	$N \times 1$	Acceptance percentages calculated for each parallel chain
Rh	$n \times 1$	Convergence index for each of the variables; see <code>mcmcci.m</code> .
Ne	$n \times 1$	Effective number of independent samples for each of the variables; see <code>mcmcci.m</code> .
AA	$M \times N \times n$	Array storing the chains: $AA(i, j, k)$ is the k th element of the parameter vector stored as the i th member of the j th chain. $AA(1, j, :) = A0(:, j)$.
IAA	$M \times N$	Acceptance indices. $IAA(i, j) = 1$ means that the proposal a^* generated at the i th step of the j th chain was accepted so that $AA(i, j, :) = a^*$. $IAA(i, j) = 0$ means that the proposal a^* generated at the i th step of the j th chain was rejected so that $AA(i, j, :) = AA(i - 1, j, :)$, $i > 1$.

3.1.2 mcmcmhic

The software component `mcmcmhic.m` has calling syntax

```
[S, aP, Rh, Ne, AA, IAA] = mcmcmhic(target, jump, M, N, M0, Q, A0)
```

This component generates N MCMC chains of length M using an independence chain algorithm. It generates proposal samples a^* according to the jumping distribution, and evaluates their acceptance probabilities defined in Equation (2). An accept-reject procedure is carried out to determine whether or not a^* is selected to be part of the MCMC sample.

The jumping distribution for the current iteration of the parameters is independent of the previous iteration in the case of independence chains. Hence `mcmcmhic.m` does not need

to evaluate the jumping probability at each iteration but merely updates these probabilities based on whether a sampled value is accepted or rejected, thereby improving computational efficiency.

It is important to note that the outputs of the jumping distribution for `mcmcmhic.m` are proposal samples and the logarithm of the jumping distribution evaluated at these samples. The outputs of the jumping distribution for `mcmcmh.m` are proposal samples and the difference $\log(p_0(a|a^*)) - \log(p_0(a^*|a))$, where p_0 is the jumping distribution.

Once the sample has been generated, they are passed to `mcmcci.m` which determines convergence indices and to `mcsums.m` which evaluates summary statistics. Details of these modules are provided in the following subsections.

	Size	Description
		Inputs
target	function handle	This function takes as input array $A(n, N)$ and returns a vector $\log p(1, N)$ where $\log p(j)$ is the logarithm of target distribution evaluated at $A(:, j)$, up to an additive constant.
jump	function handle	This function returns $A^*(n, N)$, proposed samples, and a $1 \times N$ vector $l0s$ of the logarithm of the jumping distribution evaluated at A^*
M		Length of the chains.
N		Number of chains.
M0		Integer M_0 specifying the ‘burn-in’ period. Constraint: $M > M_0 \geq 0$.
Q	$nQ \times 1$	Quantiles to be evaluated. Ranges from 0 (minimum of sample) to 100 (maximum of sample) and 50 represents the median value
A0	$n \times N$	Array of feasible starting points: the target distribution evaluated at $A0(:, j)$ is strictly positive.
		Output
S	$(2 + nQ) \times n$	summary statistics pertaining to the samples from the target distribution: mean, standard deviation and percentile limits, where the percentile limits are given by Q.
aP	$N \times 1$	Acceptance percentages calculated for each parallel chain
Rh	$n \times 1$	Convergence index for each of the variables.
Ne	$n \times 1$	Effective number of independent samples for each of the variables.
AA	$M \times N \times n$	Array storing the chains: $AA(i, j, k)$ is the k th element of the parameter vector stored as the i th member of the j th chain. $AA(1, j, :) = A0(:, j)$.
IAA	$M \times N$	Acceptance indices. $IAA(i, j) = 1$ means that the proposal a^* generated at the i th step of the j th chain was accepted so that $AA(i, j, :) = a^*$. $IAA(i, j) = 0$ means that the proposal a^* generated at the i th step of the j th chain was rejected so that $AA(i, j, :) = AA(i - 1, j, :)$, $i > 1$.

3.2 Convergence indices

The software component `mcmcci.m` has calling syntax

```
[Rhat, Neff] = mcmcci(A, M0)
```

This component may be applied to the outputs of any Markov chain Monte Carlo scheme involving multiple chains. It involves evaluating convergence indices which indicate whether the chains have converged to the target distribution based on the scheme described in Section 2.

	Size	Description
		Inputs
A	$M \times N$	Array A storing N chains of length M for a single parameter. Constraint: $N > 1$.
M0		Integer M_0 specifying the ‘burn-in’ period. Constraint: $M > M_0 \geq 0$. <i>Note:</i> the convergence indices are calculated on the basis of $A(q, r)$, $q \geq M_0 + 1$.
		Output
Rhat		Convergence index \hat{R} . In theory, $\hat{R} \geq 1$ and the closer the value is to 1, the more confidence that convergence has been achieved. The output value is $\max\{\hat{R}, 1\}$.
Neff		Effective number n_{eff} of independent samples. In theory $n_{\text{eff}} \leq (M - M_0)N$ and the closer n_{eff} is to the limit $(M - M_0)N$, the less autocorrelation in the chains. The output value is $\min\{(M - M_0)N, n_{\text{eff}}\}$.

3.3 Sample summary

The software component `mcsums.m` has calling syntax

```
[abar, s, aQ] = mcsums(A, M0, Q)
```

This component may be applied to the outputs of any Markov chain Monte Carlo scheme (or indeed any sampling scheme). It provides estimates of quantiles derived from the samples associated with the variables. Let n be the number of variables in the sample. Given $0 \leq q \leq 100$, the associated quantile Q is such that

$$\Pr(\mathbf{a} \leq Q) = q/100.$$

If $q = 0$, the component calculates the minimum of the sample; if $q = 100$, the component calculates the maximum of the sample.

	Size	Description
Inputs		
A	$M \times N$	Array: Array storing N chains of length M associated with a parameter. Constraint: $N > 1$.
M0		Integer M_0 specifying the ‘burn-in’ period. Constraint: $M > M_0 \geq 0$. <i>Note:</i> the summary information for the j th variable are calculated on the basis of $A(q, r)$, $q \geq M_0 + 1$, regarded as a vector.
Q	$n_Q \times 1$	Quantiles to be evaluated. Ranges from 0 (minimum of sample) to 100 (maximum of sample) and 50 represents the median value
Output		
abar		Sample mean: <code>abar</code> stores the mean of the sample associated with the variable.
s		Sample standard deviation: <code>s</code> stores the standard deviation of the sample associated with the variable.
aQ	$n_Q \times 1$	Estimated quantiles Q_l is an estimate of the quantile specified by q_l for the variable.

3.4 Jumping distributions

Proposed sample values \mathbf{a}^* are generated from the jumping distribution. As mentioned, they are of two broad types, random walk and independence chain. Some common jumping distributions are outlined in the following sections.

3.4.1 Gaussian random walk

The software component `jumpwrg.m` has calling syntax

```
[As, dp0] = jumpwrg(A, L)
```

It is used as a function handle for the `mcmcmh.m` routine. Samples of parameters are generated from a Gaussian distribution with samples at the current iteration of the parameter array as the mean and a fixed variance matrix. The ratio in equation (1) includes the jumping probability of moving between the current and proposed samples. `jumpwrg.m` evaluates the logarithm of this ratio. For a Gaussian random walk this is always zero.

	Size	Description
Inputs		
A	$n \times N$	The samples at the current iteration of the parameters
L	$n \times n$	Cholesky lower triangular factor of variance matrix of the parameter vector
Output		
As	$n \times N$	Proposed parameter array which is randomly sampled from the jumping distribution
dp0	$1 \times N$	The difference between the logarithm of the jumping distribution associated with moving from $a = A(:, j)$ to $a^* = A^*(:, j)$ and that associated with moving from a^* to a , up to an additive constant: $\log(p_0(a a^*)) - \log(p_0(a^* a))$.

3.4.2 Gaussian independence chain

The software component `jumpicg.m` has calling syntax

```
[As, l0] = jumpicg(A0, L)
```

It is used as a function handle for the `mcmcmhic.m` routine. Samples of parameters are generated from a Gaussian distribution with a fixed mean vector and variance matrix. `jumpicg.m` evaluates the logarithm of the jumping distribution evaluated at As.

	Size	Description
Inputs		
A0	$n \times N$	Starting parameter array; mean of the jumping distribution
L	$n \times n$	Cholesky factor of variance matrix of the parameter vector
Output		
As	$n \times N$	Proposed parameter array which is randomly sampled from the jumping distribution
l0	$1 \times N$	Logarithm of the jumping distribution evaluated at As up to an additive constant.

3.5 Target distributions

In order to evaluate the ratio in equation (1), the target distribution needs to be evaluated at the current and proposed samples. The `mcmcmh.m` and `mcmcmhic.m` modules evaluate the logarithm of the acceptance ratio and hence, the logarithm target distribution up to an additive constant needs to be evaluated.

3.5.1 tar.m

tar.m has calling syntax

```
T = tar(ad,y,s0,m0d,s0d,m0a)
```

Consider the model

$$y_i = \alpha\delta + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma_0^2), \quad i = 1, \dots, m.$$

It is assumed that σ_0 is known. α and δ are the quantities of interest and Gamma priors are assigned to them:

$$\alpha \sim G(m_{0,a}/2, m_{0,a}/2), \quad \delta \sim G(m_{0,d}/2, m_{0,d}\sigma_{0,d}^2/2).$$

The logarithm posterior distribution for $\log(\alpha)$ and $\log(\delta)$ is

$$\begin{aligned} \log(p(\log(\alpha), \log(\delta) | \mathbf{y})) &\propto -\frac{1}{2\sigma_0^2} \sum_{i=1}^m (y_i - \alpha\delta)^2 + \frac{m_{0,a}}{2} \log(\alpha) - \alpha \frac{m_{0,a}}{2} \\ &\quad + \frac{m_{0,d}}{2} \log(\delta) - \delta \frac{m_{0,d}\sigma_{0,d}^2}{2}. \end{aligned}$$

The inputs and output of this module are described below.

	Size	Description
		Inputs
ad	$2 \times N$	The samples at the current iteration of the parameters. $A(1, :)$ are samples of $\log(\alpha)$ and $A(2, :)$ are samples from $\log \delta$.
y	$m \times 1$	Vector of data points.
s0		Known standard deviation of y.
m0d		Parameter of the prior for δ
s0d		Parameter of the prior for δ
m0a		Parameter of the prior for α
		Output
T	$1 \times N$	logarithm of the posterior distribution of $\log(\alpha)$ and $\log(\delta)$ up to an additive constant.

4 Numerical example

Consider the model

$$y_i = \alpha\delta + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma_0^2), \quad i = 1, \dots, m.$$

It is assumed that σ_0 is known. α and δ are the quantities of interest and Gamma priors are assigned to them:

$$\alpha \sim G(m_{0,a}/2, m_{0,a}/2), \quad \delta \sim G(m_{0,d}/2, m_{0,d}\sigma_{0,d}^2/2).$$

The posterior distribution for α and δ is proportional to the prior times the likelihood by Bayes theorem:

$$\begin{aligned} p(\alpha, \delta | \mathbf{y}) &\propto p(\mathbf{y} | \alpha, \delta) p(\alpha, \delta), \\ &= \exp \left\{ -\frac{1}{2\sigma_0^2} \sum_{i=1}^m (y_i - \alpha\delta)^2 \right\} \alpha^{m_{0,a}/2-1} \exp \{-\alpha m_{0,a}/2\} \\ &\quad \times \delta^{m_{0,d}/2-1} \exp \{-\delta m_{0,d}\sigma_{0,d}^2/2\}. \end{aligned} \quad (3)$$

Since α and δ cannot be directly sampled from this distribution, the Metropolis-Hastings MCMC algorithm is used. A possible jumping distribution for this problem is a Gaussian random walk jumping distribution on $\log(\alpha)$ and $\log(\delta)$. The logarithm of the corresponding target distribution is

$$\begin{aligned} \log(p(\log(\alpha), \log(\delta) | \mathbf{y})) &\propto -\frac{1}{2\sigma_0^2} \sum_{i=1}^m (y_i - \alpha\delta)^2 + \frac{m_{0,a}}{2} \log(\alpha) - \alpha \frac{m_{0,a}}{2} \\ &\quad + \frac{m_{0,d}}{2} \log(\delta) - \delta \frac{m_{0,d}\sigma_{0,d}^2}{2}. \end{aligned} \quad (4)$$

The variance matrix of the Gaussian random walk jumping distribution needs to be estimated. This is done by finding the inverse of the Hessian matrix (second derivative of the negative logarithm of the posterior distribution) evaluated at the maximum a posteriori (MAP) estimates of the parameters $\log(\alpha)$ and $\log(\delta)$.

The samples obtained from the MCMCMH software would pertain to $\log(\alpha)$ and $\log(\delta)$ and those from α and δ can be obtained by taking their exponent.

Samples have been generated using both a random walk algorithm as well as an independence chain algorithm. In the case of the random walk algorithm, the mean of the Gaussian jumping distribution are samples at the current iteration of the parameter array. The mean of the Gaussian jumping distribution in the case of the independence chain algorithm is a vector consisting of the MAP estimates of $\log(\alpha)$ and $\log(\delta)$. In both cases the variance matrix is the inverse of the Hessian matrix described above.

The data vector \mathbf{y} , of length $m = 10$, has been generated from a Gaussian distribution with mean $\alpha\delta = 150 \times 0.5$ and standard deviation $\sigma_0 = 1$. The prior parameters are $m_{0,a} = 5$, $m_{0,d} = 10$ and $\sigma_{0,d} = 0.1$.

Samples of size 1100×100 were generated for each parameter. The length of the burn-in period was 100 samples. Histograms of samples obtained from the posterior distributions and plots of the prior distributions of α and δ are shown in Figure 1.

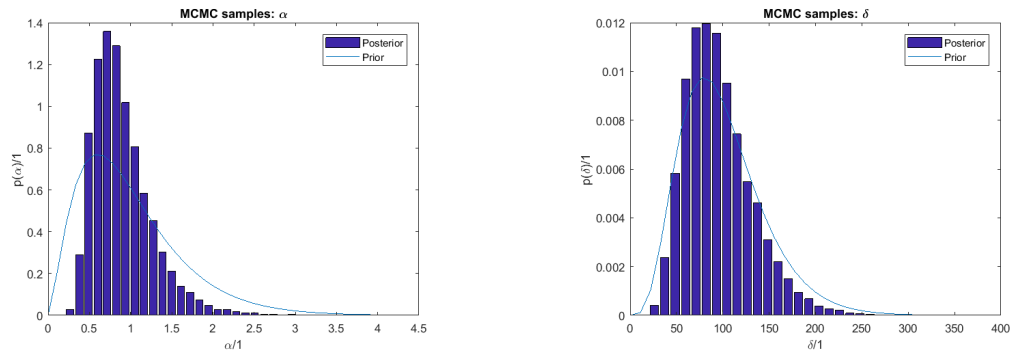


Figure 1: Prior and posterior distributions of α and δ estimated from MCMC samples.

Summary statistics and convergence indices based on the Gaussian random walk algorithm are given below for a mean acceptance of 22%.

Convergence indices

Parameter 1: 1.004218

Parameter 2: 1.004226

Effective number of independent samples

Parameter 1: 10667

Parameter 2: 10649

Summary information for posterior distribution

Mean

Parameter 1: -0.179049

Parameter 2: 4.49284

Standard deviation

Parameter 1: 0.377056

Parameter 2: 0.377069

2.5 percentile

Parameter 1: -0.884899

Parameter 2: 3.72752

Median
Parameter 1: -0.191559
Parameter 2: 4.50619

97.5 percentile
Parameter 1: 0.585931
Parameter 2: 5.19878

Summary statistics and convergence indices based on the Gaussian independence chain algorithm are given below. The mean acceptance in this case is 96.7%. Since the jumping distribution is a good approximation to the target distribution, the acceptance probability is very high. Consequently, the effective number of independent samples is also large.

Convergence indices
Parameter 1: 1.000086
Parameter 2: 1.000089

Effective number of independent samples
Parameter 1: 85330
Parameter 2: 84902

Summary information for posterior distribution

Mean
Parameter 1: -0.171457
Parameter 2: 4.48525

Standard deviation
Parameter 1: 0.380518
Parameter 2: 0.380545

2.5 percentile
Parameter 1: -0.890695
Parameter 2: 3.72463

Median
Parameter 1: -0.180745
Parameter 2: 4.49427

97.5 percentile
Parameter 1: 0.589735
Parameter 2: 5.20452

References

- [1] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, Fl., second edition, 2004.