

NPL REPORT MS 22

**USER MANUAL FOR NLLSMH: SOFTWARE IMPLEMENTING A
METROPOLIS-HASTINGS ALGORITHM FOR NON-LINEAR LEAST
SQUARES PROBLEMS**

K JAGAN AND A B FORBES

APRIL 2018

User manual for NLLSMH: Software implementing a Metropolis-Hastings algorithm for non-linear least squares problems

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. The software generates samples from a Bayesian posterior distribution for parameters of a non-linear model. The Metropolis-Hastings Markov chain Monte Carlo algorithm is used for this purpose.

NPL Report MS 22

© 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 | Mathematical Formulation | 1 |
| 2.1 | Bayesian posterior distribution for the non-linear least squares problem . . | 1 |
| 2.2 | The Metropolis-Hastings algorithm | 3 |
| 2.3 | Metropolis-Hastings sampling from the posterior distribution | 4 |
| 3 | Software implementation | 6 |
| 3.1 | Gaussian random walk | 7 |
| 3.1.1 | nllsmh_grw | 7 |
| 3.1.2 | tar_grw | 9 |
| 3.1.3 | eg_grw | 10 |
| 3.1.4 | jump_grw | 10 |
| 3.1.5 | scale_grw | 11 |
| 3.2 | Normal Gamma independence chain | 13 |
| 3.2.1 | nllsmh_ngic | 13 |
| 3.2.2 | tar_ngic | 14 |
| 3.2.3 | jump_ngic | 15 |
| 3.3 | Student's t random walk | 16 |
| 3.3.1 | nllsmh_trw | 16 |
| 3.3.2 | tar_trw | 17 |
| 3.3.3 | jump_trw | 18 |
| 3.3.4 | scale_trw | 18 |
| 3.4 | Students t independence chain | 19 |
| 3.4.1 | nllsmh_tic | 19 |
| 3.4.2 | jump_tic | 20 |
| 3.5 | Nonlinear functions | 21 |
| 3.5.1 | ED | 21 |
| 3.5.2 | CIR | 22 |
| 4 | Numerical examples | 22 |
| 4.1 | Exponential decay model | 22 |
| 4.2 | Arc fitting model | 26 |
| A | Appendix | 29 |
| A.1 | Change of variables formula | 29 |

1 Introduction

1.1 Scope

This document describes NLLSMH MATLAB software implementing the Metropolis-Hastings Markov chain Monte Carlo (MCMC) algorithm to generate samples from a Bayesian posterior distribution for parameters of a non-linear model and for the precision parameter associated with the data. Bayesian posterior distributions for non-linear models cannot be analytically specified and hence the Metropolis-Hastings MCMC algorithm is used to draw samples from the distribution.

The NLLSMH software calls NPL's MCMCMH software which generates MCMC samples from the required posterior distribution. The software also belongs to NPL's 'MCMC Software for Metrology Applications' package. The NLLSMH software also uses the Statistics and Machine Learning toolbox and the Optimisation toolbox of MATLAB.

Section 2 describes the mathematical formulation of the problem, 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 Mathematical Formulation

2.1 Bayesian posterior distribution for the non-linear least squares problem

This software concerns a model of the form

$$y_i = h_i(\boldsymbol{\alpha}) + \epsilon_i, \quad \epsilon_i \in N(0, \phi^{-1}\sigma_i^2), \quad i = 1, \dots, m.$$

Here, y_i is the measured response, $h_i(\boldsymbol{\alpha})$ the modelled response depending on parameters $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^T$, $n \leq m$, and ϵ_i represents a random effect drawn from a Gaussian sampling distribution. This sampling distribution defines the likelihood of observing \mathbf{y} given the parameters $\boldsymbol{\alpha}$ and ϕ :

$$p(\mathbf{y}|\boldsymbol{\alpha}, \phi) = p_N(\mathbf{y}|\mathbf{h}(\boldsymbol{\alpha}), \phi^{-1}\boldsymbol{\sigma}^2) \propto \phi^{m/2} \exp \left\{ -\frac{\phi}{2} \sum_{i=1}^m \left(\frac{y_i - h_i(\boldsymbol{\alpha})}{\sigma_i} \right)^2 \right\}.$$

Given that \mathbf{y} has been observed, we wish to make inferences about $\boldsymbol{\alpha}$ and ϕ .

Define $f_i(\boldsymbol{\alpha}) = (y_i - h_i(\boldsymbol{\alpha}))/\sigma_i$ and

$$F(\boldsymbol{\alpha}) = \sum_{i=1}^m f_i^2(\boldsymbol{\alpha}).$$

The Jacobian matrix J has elements

$$J_{ij} = \frac{\partial f_i}{\partial \alpha_j} \quad (1)$$

given by the derivative of f_i with respect to α_j evaluated at $\boldsymbol{\alpha} = \mathbf{a}$.

The non-linear least squares point estimate of $\boldsymbol{\alpha}$ is given by the \mathbf{a} that minimises $F(\boldsymbol{\alpha})$. If $m > n$, then a point estimate of $1/\phi$ is given by

$$\hat{\sigma}^2 = \frac{F(\mathbf{a})}{m - n}. \quad (2)$$

In a Bayesian setting (see, e.g., [2]) knowledge about $\boldsymbol{\alpha}$ and ϕ derived from \mathbf{y} is encapsulated in the posterior distribution $p(\boldsymbol{\alpha}, \phi|\mathbf{y})$ where

$$p(\boldsymbol{\alpha}, \phi|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\alpha}, \phi)p(\boldsymbol{\alpha}, \phi),$$

and $p(\boldsymbol{\alpha}, \phi)$ is the prior distribution for $\boldsymbol{\alpha}$ and ϕ . A non-informative prior for $\boldsymbol{\alpha}$ but an informative, possibly vague, prior for ϕ is

$$p(\boldsymbol{\alpha}) \propto 1, \quad \phi \sim G(m_0/2, m_0\sigma_0^2/2).$$

$\boldsymbol{\alpha}$ and ϕ are assumed to be independent a-priori, and G represents the gamma distribution.

The term σ_0^2 is the prior estimate of $1/\phi$ and $m_0 \geq 1$ is a measure of the strength of belief in this prior estimate, the larger the value of m_0 , the greater the belief.

The posterior distribution corresponding to this prior is then

$$p(\boldsymbol{\alpha}, \phi|\mathbf{y}) \propto \phi^{(m+m_0)/2-1} \exp \left\{ -\frac{\phi}{2} \left[m_0\sigma_0^2 + f(\boldsymbol{\alpha})^\top f(\boldsymbol{\alpha}) \right] \right\}, \quad (3)$$

where, $f(\boldsymbol{\alpha}) = (f_1, \dots, f_m)^\top$.

If the posterior distribution of the precision is not of interest, then equation (3) can be marginalised with respect to ϕ . The resulting marginal posterior distribution for $\boldsymbol{\alpha}$ is given by

$$p(\boldsymbol{\alpha}|\mathbf{y}) \propto \left[m_0\sigma_0^2 + f(\boldsymbol{\alpha})^\top f(\boldsymbol{\alpha}) \right]^{-(m+m_0)/2}. \quad (4)$$

2.2 The Metropolis-Hastings algorithm

For nonlinear $h_i(\alpha)$, the posterior distribution is unlikely to be given in closed form. However, we can generate samples of α and ϕ from the posterior distribution using an MCMC iterative algorithm [1, 2]. In particular, the Metropolis-Hastings MCMC algorithm can 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 only depend on the values at the previous iteration, i.e., they satisfy the Markov property. The samples must also be provided in such a way that they converge to the target distribution $p(\mathbf{a})$, where \mathbf{a} is the parameter vector. 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 draw \mathbf{a}_{q-1} , a proposed draw \mathbf{a}^* for the next member of the sequence is generated 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

$$P_q = \min\{1, r_q\}, \quad r_q = \frac{p(\mathbf{a}^*)p_0(\mathbf{a}_{q-1}|\mathbf{a}^*)}{p(\mathbf{a}_{q-1})p_0(\mathbf{a}^*|\mathbf{a}_{q-1})}. \quad (5)$$

The simplest way to implement the acceptance step is to draw u_q from the uniform distribution $R(0, 1)$ and if $u_q < P_q$, 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 the probability of jumping from \mathbf{a}_{q-1} to \mathbf{a}^* is the same as that of jumping from \mathbf{a}^* to \mathbf{a}_{q-1} . This *reversibility* property is a sufficient condition for $p(\mathbf{a})$ to be the limiting distribution of the chain. For $p(\mathbf{a})$ to be the limiting distribution of the chain, $\{a_q\}, q = 1, \dots, M$, the probability of moving between any two points in the parameter space should be non-zero as should be the probability of staying on the same point.

The important practical feature of the acceptance probability is that $p(\mathbf{a})$ and $p_0(\mathbf{a}^*|\mathbf{a})$ need only be known up to a normalising 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 which is easier to sample from. The percentage of samples that are accepted is larger if the jumping distribution is closer to the target distribution.

Test for convergence. Convergence of chains can be assessed by comparing the behaviour of chains of the same length generated using different starting points [2]. Suppose we have N chains each of length M . For each parameter, the variance between chains B and the variance within a chain W are computed. The convergence statistic \hat{R} , a function of B and W , represents the potential reduction in the estimate of the standard deviation of the posterior distribution as $M \rightarrow \infty$. It is expected that \hat{R} will approach 1 from above and a value of \hat{R} close to 1 indicates convergence.

Another measure of effectiveness of the sampling algorithm is the effective number of inde-

pendent draws n_e . In general, samples within a chain will be autocorrelated. The effective number of independent draws is bounded above by the total number of samples generated and the closer n_e is to the upper bound, the less autocorrelated are the samples.

2.3 Metropolis-Hastings sampling from the posterior distribution

The software is divided into four parts; each part implements a different jumping distribution. Jumping distributions are broadly of two types, random walk and independence chain. While random walk jumping distributions draw samples that are dependent on those at the previous iteration, independence chain jumping distributions draw samples that are independent of past iterations. Samples from the target distributions in Equations (3) and (4) are obtained using both random walk and independence chain schemes. In each case, the target and suitable jumping distribution are specified and the MCMCMH software is used to draw samples.

Gaussian random walk

Samples of α and $\log(\phi)$ are obtained from the posterior distribution

$$p(\alpha, \log(\phi)|\mathbf{y}) \propto \phi^{(m+m_0)/2} \exp \left\{ -\frac{\phi}{2} \left[m_0 \sigma_0^2 + f(\alpha)^\top f(\alpha) \right] \right\}. \quad (6)$$

This distribution is obtained using the change of variables formula in Appendix A.1. The jumping distribution is a Gaussian distribution with the sample from the previous iteration of the Metropolis-Hastings algorithm as the mean and a fixed variance matrix given by the inverse of the Hessian matrix (second derivative of the target distribution with respect to the parameters) evaluated at the maximum-a-posteriori (MAP) estimates of the parameters. The software uses MATLAB's `fminunc` function in the Optimisation toolbox in order to evaluate the MAP estimates.

Normal gamma independence chain

The normal gamma jumping distribution for α and ϕ is derived by linearising $f(\alpha)$ around its non-linear least squares estimate $f(\mathbf{a})$. The first order Taylor series approximation of $f(\alpha)$ is given by $f(\alpha) \approx f(\mathbf{a}) + J(\alpha - \mathbf{a})$, where J is the Jacobian matrix evaluated at $\alpha = \mathbf{a}$. Substituting the above in equation (3), we obtain the approximate posterior density

$$p(\alpha, \phi|\mathbf{y}) \propto \phi^{(m+m_0)/2-1} \exp \left\{ -\frac{\phi}{2} \left[m_0 \sigma_0^2 + (m-n)\hat{\sigma}^2 + (\alpha - \mathbf{a})^\top J^\top J(\alpha - \mathbf{a}) \right] \right\}, \quad (7)$$

where $(m - n)\hat{\sigma}^2 = f(\mathbf{a})^\top f(\mathbf{a})$.

Using the theory of conditional probability equation (7) can be decomposed into $p(\boldsymbol{\alpha}, \phi | \mathbf{y}) \propto p(\phi | \mathbf{y})p(\boldsymbol{\alpha} | \phi, \mathbf{y})$. Since $\boldsymbol{\alpha}$ appears linearly in Equation (7), $p(\phi | \mathbf{y})$ and $p(\boldsymbol{\alpha} | \phi, \mathbf{y})$ are evaluated analytically to be

$$\phi | \mathbf{y} \sim G(\nu/2, \nu\bar{\sigma}^2/2), \quad \boldsymbol{\alpha} | \phi, \mathbf{y} \sim N(\mathbf{a}, \phi^{-1}(J^\top J)^{-1}), \quad (8)$$

where $\nu = m + m_0 - n$ and $\bar{\sigma}^2 = (m_0\sigma_0^2 + (m - n)\hat{\sigma}^2)/\nu$.

The approximate Normal-Gamma posterior distribution in equation (8) can be used as a jumping distribution to sample from the posterior distribution in equation (3). This algorithm uses MATLAB's `gamrnd` function in the Statistics and Machine Learning toolbox to generate jumping samples of the precision parameter.

Multivariate t random walk

If the posterior distribution for the precision ϕ is not of interest, samples from the marginal distribution for $\boldsymbol{\alpha}$ in Equation (4) can be generated. The linearised posterior distribution in Equation (7) can be marginalised analytically with respect to ϕ :

$$\boldsymbol{\alpha} | \mathbf{y} \sim t_\nu(\mathbf{a}, \bar{\sigma}^2(J^\top J)^{-1}). \quad (9)$$

Jumping samples are obtained from this t distribution with mean equal to the sample from the previous iteration of the MH algorithm, ν degrees of freedom and scaling $\bar{\sigma}^2(J^\top J)^{-1}$. This algorithm uses MATLAB's `mvt rnd` function in the Statistics and Machine Learning toolbox to generate jumping samples of $\boldsymbol{\alpha}$.

Multivariate t independence chain

This algorithm samples from the marginal distribution of $\boldsymbol{\alpha}$ in Equation (4) with jumping distribution in Equation (9). The mean of the jumping distribution in this case is the non-linear least squares estimate of model parameter.

This algorithm uses MATLAB's `mvt rnd` function in the Statistics and Machine Learning toolbox to generate jumping samples of $\boldsymbol{\alpha}$.

A note on the various sampling schemes

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 draws. 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 draw only 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 jumping distribution generates samples that are dependent on the samples 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 MH 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 should yield a higher acceptance rate than a random walk scheme.

3 Software implementation

The NLLSMH software is implemented through several MATLAB modules. The parent modules are `nllsmh_grw`, `nllsmh_trw`, `nllsmh_ngic` and `nllsmh_tic` which specify the target and jumping distributions that implement the Gaussian random walk (GRW), Student's t random walk (TRW), Normal-Gamma independence chain (NGIC) and Student's t independent chain (TIC) algorithms respectively. They call NPL's MCMCMH software to draw samples from the target distribution using the Metropolis-Hastings MCMC algorithm.

The two random walk algorithms provide the user with the option to scale the jumping distribution in such a way that the acceptance probabilities are approximately a specified value. A scale factor multiplies the variance matrix of the jumping distribution that regulates its spread. The jumping samples should tend to visit high probability parts of the parameter space but also explore the space without getting stuck at one point. A scale factor greater than one could lead to faster convergence, as the jumping distribution would scan the support of the parameter space faster, at the price of lower acceptance probabilities. A scale factor less than one could lead to slower convergence, however the acceptance probability would be higher.

In order to determine a reasonable scale factor, a set of scale factor values is considered and MCMC samples are drawn for each value using a small chain length. The logarithm of the acceptance probability for each problem is computed and a straight line model with the scale factor determined as the dependent variable and logarithm acceptance as dependent variable determined. This model can then be used to evaluate the factor needed to achieve some acceptance probability. The algorithm is outlined below.

Given a vector of scale factors s of length m_s , and the desired acceptance probability t_a ,

```

for  $i = 1, \dots, m_s$  do
    I. Set the variance matrix of the jumping distribution scaled by  $s(i)$ .
    II. Generate a Metropolis Hastings sample of modest length (say  $M = 200$ )
        from the target distribution and evaluate acceptance probabilities  $aP$ .
    III. Store the logarithm of the mean of  $aP$  as  $p_{a,i}$ .
end

```

Perform a least squares fit the linear model $s = b_1 + b_2 p_a$ to obtain estimates \hat{b}_1 and \hat{b}_2 .

Evaluate the scale sc associated with the desired acceptance probability t_a using the fitted model $sc = \hat{b}_1 + \hat{b}_2 \log(t_a)$.

Algorithm 1: scale factor determination

A scale factor for the two independence chain algorithms needs to be input by the user as there is no recommended acceptance probability for these. An over-disposed jumping distribution i.e. $sc \geq 1$ is recommended.

An exponential decay model and a circle fitting model are considered to illustrate the use of the software. ED evaluates the exponential decay function and its Jacobian matrix and CIR evaluates the function and Jacobian for the circle fitting problem.

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

3.1 Gaussian random walk

3.1.1 nllsmh_grw

The software component `nllsmh_grw.m` has calling syntax

```
[S, aP, Rh, Ne, AA, IAA, sc] = nllsmh_grw(ahat, fun, m0, s0, M, N, M0, Q, I)
```

This module sets up the jumping and target distributions and calls `mcmcmh.m` to generate samples using the Metropolis-Hastings algorithm. In this case, the target distribution is the logarithm of the posterior distribution of α and $\log(\phi)$ given in Equation (6) and a Gaussian random walk jumping distribution is used to draw proposal samples. The variance matrix of the Gaussian random walk jumping distribution is the inverse of the Hessian matrix evaluated at the MAP estimates of α and $\log(\phi)$. The jumping distribution can be scaled to make it more or less dispersed. The scale factor can either be set by the user or evaluated to obtain some level of acceptance. The choice is governed by logical input I which can be

either `true` or `false`. The algorithm is outlined below.

Given f_i as described in section 2, non-linear least squares estimates of model parameters $\hat{\alpha}$ and prior parameters m_0 and s_0

I. Assign the target distribution `tar_grw.m`.

II. Find an estimate of the variance matrix of the Gaussian jumping distribution.

1. Set starting points for optimisation as a_0 a vector containing $\hat{\alpha}$ and $\log(1/s_0^2)$.
2. Evaluate the maximum-a-posteriori estimates of the parameters $\check{\alpha}$ and $\log(\phi)$ using an unconstrained optimisation routine `fminunc` with a_0 as the starting point. This computation requires the energy (negative logarithm of the posterior distribution) and gradient (derivative of the energy with respect to the parameters) which are computed in `eg_grw`.
3. Evaluate the Hessian, H , at $\check{\alpha}$ and $\log(\phi)$ using finite differences.
4. Set the variance matrix, V , of α and $\log(\phi)$ to be the inverse of the H .

III. Set starting points for MCMC: Generate random starting points from a Gaussian distribution with mean $\check{\alpha}$ and variance V .

IV. Set scale factor `sc`: The scale factor can either be set by the user or computed to yield a certain acceptance probability.

if true then

| User prompted to input scale

else if false then

| User prompted to input acceptance probability and optimum scale computed based on Algorithm 1

end

V. Gaussian random walk jumping distribution defined with variance V and scale `sc`. Random samples are drawn with the previous state of the parameter array as the mean. The Cholesky factor of the variance matrix L is computed and scaled by `sc`.

VI. MCMCMH software is called to generate samples and compute convergence and summary statistics.

Algorithm 2: `nllsmh_grw` algorithm

The inputs and outputs to this module are described below.

| | Size | Description |
|------|-----------------------|---|
| | | Inputs |
| ahat | $n \times 1$ | Nonlinear least squares estimates of parameters α of the model |
| fun | function handle | Takes as input α values and returns function values $f(m, 1)$ and Jacobian $J(m, n)$ for given α where, $f_i = (y_i - h_i(\alpha))/\sigma_i$, $i = 1, \dots, m$ and J is defined in Equation (1). |
| m0 | | Prior degree of belief on estimate of standard deviation of the data. If σ_0 is determined from a set of repeated experiments, then m_0 can be set to the number of experiments. |
| s0 | | Prior estimate of standard deviation of the data. If the estimates of σ_i are reasonable, then σ_0 is usually set to 1 indicating that the prior distribution for ϕ has a mean of 1. |
| M | integer | Length of MCMC chains |
| N | integer | Number of MCMC chains |
| M0 | integer | Length of the burn in period |
| 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 |
| I | true or false | Logical variable, if I is true, then a scale factor for the jumping distribution has to be input. If I is false, then desired acceptance probability is to be input and a corresponding scale is determined using Algorithm 1. |
| | | Outputs |
| 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 chain |
| Rh | $n \times 1$ | Convergence index for each of the parameters. |
| Ne | $n \times 1$ | Effective number of independent draws for each of the variables. |
| AA | $M \times N \times n$ | Array storing the chains: $AA(i, j, k)$ is the kth element of the parameter vector stored as the ith member of the jth chain. $AA(1, j, :) = A0(:, j)$. |
| IAA | $M \times N$ | Acceptance indices. $IAA(i, j) = 1$ means that the proposal a^* generated at the ith step of the jth chain was accepted so that $AA(i, j, :) = a^*$. $IAA(i, j) = 0$ means that the proposal a^* generated at the ith step of the jth chain was rejected so that $AA(i, j, :) = AA(i - 1, j, :)$, $i > 1$. |
| sc | | The scale factor multiplied to the Cholesky factor of the variance matrix of the jumping distribution. |

3.1.2 tar_grw

The software component `tar_grw.m` has calling syntax

```
T = tar_grw(alp, fun, m0, s0)
```

This function returns the logarithm of the posterior distribution of α and $\log(\phi)$ in Equation (6).

The inputs and outputs to this module are described below.

| | Size | Description |
|----------------------------|---------------------------------|--|
| | | Inputs |
| alp fun m0 s0 | $p \times N$ function handle | Current state of parameter array for N chains, $n = p - 1$ is the length of α Takes as input α values and returns function values $f(m, 1)$ and Jacobian $J(m, n)$ for given α where, $f_i = (y_i - h_i(\alpha))/\sigma_i$ and J is the derivative of f with respect to α . Prior degree of belief on estimate of standard deviation of the data Prior estimate of standard deviation of the data |
| | | Output |
| T | $1 \times N$ | logarithm of the posterior distribution of α and $\log(\phi)$ determined up to an additive constant |

3.1.3 eg_grw

The software component `eg_grw.m` has calling syntax

```
[E,G] = eg_grw(alp,fun,tar,m0,s0)
```

This module computes the energy i.e. the negative logarithm of the target distribution, along with its analytical gradient. This is used to find MAP estimates of the parameters and hence construct a Hessian matrix to approximate the variance matrix of the Gaussian random walk jumping distribution in the case where samples are drawn from the posterior distribution of α and $\log(\phi)$.

The inputs and outputs to this module are described below.

| | Size | Description |
|----------------------------|---------------------------------|--|
| | | Inputs |
| alp fun m0 s0 | $p \times N$ function handle | Current state of parameter array for N chains, $n = p - 1$ is the length of α Takes as input α values and returns function values $f(m, 1)$ and Jacobian $J(m, n)$ for given α where, $f_i = (y_i - h_i(\alpha))/\sigma_i$ and J is the derivative of f with respect to α . Prior degree of belief on estimate of standard deviation of the data Prior estimate of standard deviation of the data |
| | | Output |
| E G | $p \times 1$ | Energy - the negative logarithm of the posterior distribution calculated at the current state of the parameter array, α and $\log(\phi)$ Gradient - The derivative of the energy with respect to each parameter evaluated at the current state of the parameter array, α and $\log(\phi)$ |

3.1.4 jump_grw

The software component `jump_grw.m` has calling syntax


```
[As, del] = jump_grw(A, L)
```

It is used as a function handle within the `mcmc_mh.m` routine. Samples of parameters are generated from a Gaussian distribution with the current state of the parameter array as the mean and a fixed variance matrix. The ratio in equation (5) includes the jumping probability of moving between the current and proposed states. `jump_grw.m` evaluates the logarithm of this ratio. For a Gaussian random walk (in fact any symmetric distribution) this is always zero.

| | Size | Description |
|--------|--------------|---|
| Inputs | | |
| A | $p \times N$ | The current state of parameter array for N chains |
| L | $p \times p$ | Cholesky factor of variance matrix of the parameter vector |
| Output | | |
| As | $p \times N$ | Proposed parameter array which is randomly sampled from the jumping distribution |
| del | $1 \times N$ | The difference between the logarithm of the jumping distribution associated with moving from a to a^* and that associated with moving from a^* to a , up to an additive constant. $\log(p_0(a a^*)) - \log(p_0(a^* a))$: for a Gaussian random walk this is always zero. |

3.1.5 `scale_grw`

The software component `scale_grw.m` has calling syntax

```
[sc, res] = scale_grw(s, tar, A0, L, ta, M, N, M0)
```

This function evaluates a scale factor which results in an acceptance probability of approximately t_a . A set of scale factor values s is considered and Metropolis-Hastings samples are drawn for a small chain length, say $M = 200$, $N = 5$ and $M_0 = 50$. The logarithm of the acceptance probability p_a for each problem is computed. A straight line model $s = b_1 + b_2 p_a$ is determined. The output sc is obtained by setting the acceptance probability at t_a in the model. The procedure is outlined in Algorithm 1.

The inputs and outputs to the function are described below.

| | Size | Description |
|-----|-----------------|---|
| | | Inputs |
| s | $ms \times 1$ | Some possible scale factors to estimate the straight line model. |
| tar | function handle | Target distribution, in this case the posterior distribution of α and $\log(\phi)$. |
| A0 | $p \times N$ | Starting point for each MCMC problem. |
| L | $p \times p$ | The Cholesky factor of the variance matrix of α and $\log(\phi)$. |
| ta | Percentage | Desired acceptance percentage. |
| M | | The length of the MCMC chains for each MCMC problem. |
| N | | The number of parallel chains for each MCMC problem. |
| M0 | | The burn-in length for each MCMC problem. |
| | | Output |
| sc | | The scale factor which results in an acceptance probability of approximately t_a . |
| res | $ms \times 1$ | Residuals evaluated from the model fit. |

3.2 Normal Gamma independence chain

3.2.1 `nllsmh_ngic`

The software component `nllsmh_ngic.m` has calling syntax

```
[S,aP,Rh,Ne,AA,IAA] = nllsmh_ngic(ahat,fun,m0,s0,sc,M,N,M0,Q)
```

This module sets up the jumping and target distribution and calls `mcmcmhic.m` to generate samples using the Metropolis-Hastings algorithm. In this case, the target distribution is the logarithm of the posterior distribution of α and ϕ in Equation (3) and a normal gamma distribution is used to draw proposal samples. If $f(\alpha)$ is linearised around the NLLS estimate of α , then the resulting posterior distribution can be decomposed into the product of a Gamma and Gaussian distribution as $p(\alpha, \phi | \mathbf{y}) \propto p(\phi | \mathbf{y})p(\alpha | \phi, \mathbf{y})$ as in Equation (8). The algorithm is outlined below.

Given f_i as described in section 2, non-linear least squares estimates of model parameters $\hat{\mathbf{a}}$ and prior parameters m_0 and s_0

- I. Evaluate the non-linear function and Jacobian J at $\hat{\mathbf{a}}$.
- II. Evaluate $\hat{\sigma} = ||f||/\sqrt{m-n}$, where m is the number of data points and n is the number of parameters.
- III. Compute $\nu = m + m_0 - n$ and $\bar{\sigma}^2 = (m_0 s_0^2 + (m-n)\hat{\sigma}^2)/\nu$.
- IV. Find the upper triangular matrix R associated with the QR factorisation of J .
- V. Define the jumping distribution using $\hat{\mathbf{a}}$, ν , $\bar{\sigma}^2$ and R and call the function to obtain initial values for the Metropolis-Hastings algorithm.
- VI. Jumping samples are obtained by first drawing ϕ_q from a Gamma distribution $G(\nu/2, \nu\bar{\sigma}^2/2)$.
- VII. Samples \mathbf{a}_q are obtained from a Gaussian distribution with mean equal to the samples in the previous iteration and variance $(R^\top R)^{-1}/\phi_q$. ν and R are divided by the scale sc . If sc is greater than one, both distributions become more dispersed.
- VIII. Define the target distribution in equation (3).
- IX. MCMCMH software is called to generate samples and compute convergence and summary statistics.

Algorithm 3: `nllsmh_ngic` algorithm

The inputs and outputs to this module are described below.

| | Size | Description |
|---------|-----------------------|---|
| Inputs | | |
| ahat | $n \times 1$ | Nonlinear least squares estimates of parameters α of the model |
| fun | function handle | Takes as input α values and returns function values $f(m, 1)$ and Jacobian $J(m, n)$ for given alpha where, $f_i = (y_i - h_i(\alpha))/\sigma_i$ and J is the derivative of f with respect to α . |
| m0 | | Prior degree of belief on estimate of standard deviation of the data |
| s0 | | Prior estimate of standard deviation of the data. If the estimates of σ_i are reasonable, then $s0$ is usually set to 1 indicating that the prior distribution for ϕ has a mean of 1. |
| sc | | Scale factor for the jumping distribution. |
| M | integer | Length of MCMC chains |
| N | integer | Number of MCMC chains |
| M0 | integer | Length of the burn in period |
| 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 |
| Outputs | | |
| 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 chain |
| Rh | $n \times 1$ | Convergence index for each of the variables. |
| Ne | $n \times 1$ | Effective number of independent draws for each of the variables. |
| AA | $M \times N \times n$ | Array storing the chains: $AA(i, j, k)$ is the kth element of the parameter vector stored as the ith member of the jth chain. $AA(1, j, :) = A0(:, j)$. |
| IAA | $M \times N$ | Acceptance indices. $IAA(i, j) = 1$ means that the proposal a^* generated at the ith step of the jth chain was accepted so that $AA(i, j, :) = a^*$. $IAA(i, j) = 0$ means that the proposal a^* generated at the ith step of the jth chain was rejected so that $AA(i, j, :) = AA(i - 1, j, :)$, $i > 1$. |

3.2.2 tar_ngic

The software component `tar_ngic.m` has calling syntax

```
T = tar_ngic(ap, fun, m0, s0)
```

This function returns the logarithm of the posterior distribution of α and ϕ as in Equation (3).

The inputs and outputs to this module are described below.

| | Size | Description |
|---------------------------|---------------------------------|--|
| | | Inputs |
| ap fun m0 s0 | $p \times N$ function handle | Current state of parameter array for N parallel chains Takes as input α values and returns function values $f(m, 1)$ and Jacobian $J(m, n)$ for given α where, $f_i = (y_i - h_i(\alpha))/\sigma_i$ and J is the derivative of f with respect to α . Prior degrees of freedom Prior estimate of standard deviation of the data |
| | | Output |
| T | $1 \times N$ | logarithm of the posterior distribution of α and ϕ determined up to an additive constant |

3.2.3 jump_ngic

The software component `jump_ngic.m` has calling syntax

```
[As, l0s] = jump_ngic(ahat, nu, s2bar, R, N)
```

It is used as a function handle within the `mcmcshic.m` routine. Samples of ϕ are generated from a Gamma distribution and these are in turn used to sample α from a Gaussian distribution with precision parameter ϕ . The parameters of the distribution are outlined in Section 2.3. The jumping density of the sample generated is also evaluated.

| | Size | Description |
|-------------------------------|--------------------------------------|--|
| | | Inputs |
| ahat nu s2bar R N | $n \times 1$ $n \times n$ | Nonlinear least squares estimates of parameters. Degrees of freedom of the multivariate t distribution Weighted mean of prior variance and variance estimated from the data. Upper triangular matrix which represents the QR factorisation of the Jacobian J . Number of chains for MCMC |
| | | Output |
| As l0s | $p \times N$ $1 \times N$ | Proposed parameter array which is randomly sampled from the jumping distribution Jumping density evaluated at the proposed parameter array. |

3.3 Student's t random walk

3.3.1 nllsmh_trw

The software component `nllsmh_trw.m` has calling syntax

```
[S, aP, Rh, Ne, AA, IAA, sc] = nllsmh_trw(ahat, fun, m0, s0, M, N, M0, Q, I)
```

This module sets up the jumping and target distribution and calls `mcmcmh.m` to generate samples using the Metropolis-Hastings algorithm. In this case, the target distribution is the logarithm of the posterior distribution of α in Equation (4) and a Multivariate t random walk jumping distribution is used to draw proposal samples. The jumping distribution can be scaled to make it more or less dispersed. The scale factor can either be set by the user or evaluated to obtain some level of acceptance. This is governed by logical input I which can be either `true` or `false`. The algorithm is outlined below.

Given f_i as described in section 2, non-linear least squares estimates of model parameters $\hat{\mathbf{a}}$ and prior parameters m_0 and s_0

I. Evaluate the non-linear function f and Jacobian J at $\hat{\mathbf{a}}$.

II. Evaluate $\hat{\sigma} = \|f\|/\sqrt{m-n}$, where m is the number of data points and n is the number of parameters.

III. Compute $\nu = m + m_0 - n$ and $\bar{\sigma}^2 = (m_0 s_0^2 + (m-n)\hat{\sigma}^2)/\nu$.

IV. Find the upper triangular matrix R associated with the QR factorisation of J .

V. Randomly generate initial samples for the MCMC algorithm from a t distribution with ν degrees of freedom, mean $\hat{\mathbf{a}}$ and covariance matrix $(R^\top R)^{-1}$.

VI. Define the target distribution in Equation 4.

VII. Set scale factor sc : The scale factor can either be set by the user or computed to yield a certain acceptance probability.

if true then

 | User prompted to input scale

else if false then

 | User prompted to input acceptance probability and optimum scale computed
 | based on Algorithm 1

end

VIII. Define jumping distribution in Equation 9 with the appropriate scale sc .

Random samples are drawn from a t distribution with the previous state of the parameter array as the mean and ν degrees of freedom. R is divided by sc , so a value $sc > 1$ would make the jumping distribution more dispersed.

IX. MCMCMH software is called to generate samples and compute convergence and summary statistics.

Algorithm 4: `nllsmh_trw` algorithm

The inputs and outputs to this module are described below.

| | Size | Description |
|------|-----------------------|---|
| | | Inputs |
| ahat | $n \times 1$ | Nonlinear least squares estimates of parameters α of the model |
| fun | function handle | Takes as input α values and returns function values $f(m, 1)$ and Jacobian $J(m, n)$ for given alpha where, $f_i = (y_i - h_i(\alpha))/\sigma_i$ and J is the derivative of f with respect to α . |
| m0 | | Prior degree of belief on estimate of standard deviation of the data |
| s0 | | Prior estimate of standard deviation of the data. This is usually set to 1 indicating that the prior distribution for ϕ has a mean of 1. |
| M | integer | Length of MCMC chains |
| N | integer | Number of MCMC chains |
| M0 | integer | Length of the burn in period |
| 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 |
| I | true or false | Logical variable, if I is true, then a scale factor for the jumping distribution has to be input. If I is false, then the scale is automatically determined such that the acceptance probability is around t_a . |
| | | Outputs |
| 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. |
| Ne | $n \times 1$ | Effective number of independent draws for each of the variables. |
| AA | $M \times N \times n$ | Array storing the chains: $AA(i, j, k)$ is the kth element of the parameter vector stored as the ith member of the jth chain. $AA(1, j, :) = A0(:, j)$. |
| IAA | $M \times N$ | Acceptance indices. $IAA(i, j) = 1$ means that the proposal a^* generated at the ith step of the jth chain was accepted so that $AA(i, j, :) = a^*$. $IAA(i, j) = 0$ means that the proposal a^* generated at the ith step of the jth chain was rejected so that $AA(i, j, :) = AA(i - 1, j, :)$, $i > 1$. |
| sc | | The scale factor multiplied to the variance matrix of the jumping distribution. |

3.3.2 tar_trw

The software component `tar_trw.m` has calling syntax

```
T = tar_trw(a, fun, m0, s0)
```

This function returns the logarithm of the marginal posterior distribution of α as in Equation (4).

The inputs and outputs to this module are described below.

| | Size | Description |
|-----|-----------------|---|
| | | Inputs |
| a | $n \times N$ | Current state of parameter array for N parallel chains |
| fun | function handle | Takes as input α values and returns function values $f(m, 1)$ and Jacobian $J(m, n)$ for given alpha where, $f_i = (y_i - h_i(\alpha))/\sigma_i$ and J is the derivative of f with respect to α . |
| m0 | | Prior degrees of freedom |
| s0 | | Prior estimate of standard deviation of the data |
| | | Output |
| T | $1 \times N$ | logarithm of the posterior distribution of α up to an additive constant |

3.3.3 `jump_trw`

The software component `jump_trw.m` has calling syntax

```
[As, del] = jump_trw(A, R, nu)
```

It is used as a function handle within the `mcmcmh.m` routine. Samples of parameters are generated from a Students t distribution with the current state of the parameter array as the mean and a fixed variance matrix and degrees of freedom. The ratio in equation (5) includes the jumping probability of moving between the current and proposed states. `jump_trw.m` evaluates the logarithm of this ratio. Since the t distribution is symmetric, this will always be zero.

| | Size | Description |
|-----|--------------|---|
| | | Inputs |
| A | $n \times N$ | The current state of parameter array |
| R | $n \times n$ | QR factorisation of the Jacobian J . The inverse of R is equivalent to the Cholesky factor of the variance matrix. |
| nu | | Degrees of freedom of the multivariate t distribution |
| | | Output |
| As | $n \times N$ | Proposed parameter array which is randomly sampled from the jumping distribution |
| del | $1 \times N$ | The difference between the logarithm of the jumping distribution associated with moving from a to a^* and that associated with moving from a^* to a , up to an additive constant. $\log(p_0(a a^*)) - \log(p_0(a^* a))$: it is always zero for a t distribution. |

3.3.4 `scale_trw`

The software component `scale_trw.m` has calling syntax


```
[sc,res] = scale_trw(s,tar,A0,R,nu,ta,M,N,M0)
```

This function is similar to `scale_grw.m` but considers R the QR factorisation of the Jacobian J and ν the degrees of freedom of the Student's t jumping distribution as inputs.

3.4 Students t independence chain

The target distribution for the independence chain algorithm is the same as the target distribution for the t random walk algorithm `tar_trw`.

3.4.1 `nllsmh_tic`

The software component `nllsmh_tic.m` has calling syntax

```
[S,aP,Rh,Ne,AA,IAA] = nllsmh_tic(ahat,fun,m0,s0,sc,M,N,M0,Q)
```

This module sets up the jumping and target distribution and calls `mcmcmhic.m` to generate samples using the Metropolis-Hastings algorithm. In this case, the target distribution is the logarithm of the posterior distribution of α in Equation (4) and a Multivariate t independence chain jumping distribution is used to draw proposal samples. The only difference between this algorithm and `nllsmh_trw` is that jumping samples are independent of each other in this case as the mean of the t jumping distribution is constant and equal to the non-linear least squares estimates of the parameters.

The inputs and outputs to this module are described below.

| | Size | Description |
|---------|-----------------------|---|
| Inputs | | |
| ahat | $n \times 1$ | Nonlinear least squares estimates of parameters α of the model |
| fun | function handle | Takes as input α values and returns function values $f(m, 1)$ and Jacobian $J(m, n)$ for given alpha where, $f_i = (y_i - h_i(\alpha))/\sigma_i$ and J is the derivative of f with respect to α . |
| m0 | | Prior degree of belief on estimate of standard deviation of the data |
| s0 | | Prior estimate of standard deviation of the data. This is usually set to 1 indicating that the prior distribution for ϕ has a mean of 1. |
| sc | | A scale factor is multiplied to the variance matrix of the jumping distribution. |
| M | integer | Length of MCMC chains |
| N | integer | Number of MCMC chains |
| M0 | integer | Length of the burn in period |
| 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 |
| Outputs | | |
| 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. |
| Ne | $n \times 1$ | Effective number of independent draws for each of the variables. |
| AA | $M \times N \times n$ | Array storing the chains: $AA(i, j, k)$ is the kth element of the parameter vector stored as the ith member of the jth chain. $AA(1, j, :) = A0(:, j)$. |
| IAA | $M \times N$ | Acceptance indices. $IAA(i, j) = 1$ means that the proposal a^* generated at the ith step of the jth chain was accepted so that $AA(i, j, :) = a^*$. $IAA(i, j) = 0$ means that the proposal a^* generated at the ith step of the jth chain was rejected so that $AA(i, j, :) = AA(i - 1, j, :)$, $i > 1$. |

3.4.2 jump_tic

The software component `jump_tic.m` has calling syntax

```
[As, l0s] = jump_tic(ahat, R, nu, N)
```

It is used as a function handle within the `mcmcmbic.m` routine. Samples of parameters are generated from a t-distribution with the least squares estimates of the parameter array as the mean and a fixed variance matrix and degrees of freedom. The logarithm of the jumping density is evaluated for this proposal sample.

| | Size | Description |
|--------|--------------|--|
| Inputs | | |
| ahat | $n \times 1$ | The nonlinear least squares estimates of the model parameters. |
| R | $n \times n$ | QR factorisation of the Jacobian J . The inverse of R is equivalent to the Cholesky factor of the variance matrix. |
| nu | | Degrees of freedom of the multivariate t distribution |
| N | | Number of chains |
| Output | | |
| As | $n \times N$ | Proposed parameter array which is randomly sampled from the jumping distribution |
| logS | $1 \times N$ | logarithm of the jumping density evaluated at As. |

3.5 Nonlinear functions

3.5.1 ED

The software component `ED.m` has calling syntax

```
[f,J] = ED(a,x,x0,y,si)
```

The exponential decay model is given by

$$h_i = \alpha_1 \exp\{-\alpha_2(x_i - x_0)\} + \alpha_3,$$

where $\alpha_1, \alpha_2, \alpha_3$ are unknown quantities.

The outputs of the model are $f_i = (y_i - h_i(\alpha))/\sigma_i$ and J , the Jacobian matrix of f with respect to α .

The inputs and outputs to this module are described below.

| | Size | Description |
|--------|--------------|---|
| Inputs | | |
| a | 3×1 | Values of parameters α for the exponential decay model |
| x | $m \times 1$ | Vector of independent variable |
| x0 | | Perturbation of x |
| y | $m \times 1$ | Vector of dependent variable, e.g. counts |
| si | $m \times 1$ | Standard deviation for each data point |
| Output | | |
| f | $m \times 1$ | Function values |
| J | $m \times 3$ | Jacobian matrix of f with respect to α . |

3.5.2 CIR

The software component `CIR.m` has calling syntax

```
[f, J] = CIR(a, X, si)
```

The outputs of this module are

$$f_i = \alpha_3 - \sqrt{(x_i - \alpha_1)^2 + (y_i - \alpha_2)^2}$$

where $\alpha_1, \alpha_2, \alpha_3$ are unknown quantities.

The outputs of the model are $f_i = (y_i - h_i(\alpha))/\sigma_i$ and J , the Jacobian matrix of f with respect to α . α_1 is the abscissa of the center of the circle, α_2 is the ordinate of the center of the circle and α_3 is the radius of the circle.

The inputs and outputs to this module are described below.

| | Size | Description |
|--------|--------------|--|
| Inputs | | |
| a | 3×1 | Parameter estimates for the circle fitting model |
| X | $m \times 2$ | Matrix storing the abscissa values in the first column and ordinate values in the second |
| si | $m \times 1$ | Standard deviation for each data point |
| Output | | |
| f | $m \times 1$ | Function values |
| J | $m \times 3$ | Jacobian matrix of f with respect to α . |

4 Numerical examples

4.1 Exponential decay model

The exponential decay model can be expressed as

$$y_i = \alpha_1 \exp\{-\alpha_2(x_i - x_0)\} + \alpha_3 + \epsilon_i, \quad \epsilon_i \sim N(0, \phi^{-1}\sigma_i^2),$$

where $\alpha_1, \alpha_2, \alpha_3$ and ϕ are unknown quantities and x_0 is the initial value for x . Such models can be used to represent rate of decay of a radioactive element over time. In this case, x is the time variable, x_0 is the time at which the first measurement was made and y stores the number of atoms present.

The posterior distributions given by Equations (4) and (6) from the exponential decay model are obtained by substituting $f_i = (y_i - h_i(\alpha))/\sigma_i, i = 1, \dots, m$, where

$$h_i(\alpha) = \alpha_1 \exp\{-\alpha_2(x_i - x_0)\} + \alpha_3.$$

Nonlinear least squares estimates of α are obtained using the Gauss Newton algorithm and input to the `nllsmh` modules. Since the target distribution is the same for all four methods, we expect histograms of samples from the various methods to be the same. In order to test this, the `mcmcci.m` module in the MCMCMH software package is used. This module calculates the convergence indices of MCMC chains, \hat{R} . Chains of the same parameter obtained using different methods can be compared as if they were samples from different chains. If the \hat{R} value obtained is close to one, we can conclude that the samples from the different methods do in fact arise from the same target distribution. The \hat{R} values are reported below. The \hat{R} values are close to one which provides evidence that the samples are generated from the same target distribution.

```
Rhat alpha
Parameter 1:      1.000144
Parameter 2:      1.000176
Parameter 3:      1.000218

Rhat phi:         1.000066
```

The convergence indices from each of the four algorithms are displayed below. Parameters 1-3 are the α parameters and parameter 4 is the ϕ parameter. All the convergence indices are close to one indicating that the chains have converged.

```
Convergence indices:GRW
Parameter 1:      1.001378
Parameter 2:      1.000908
Parameter 3:      1.001401
Parameter 4:      1.000433
```

```
Convergence indices:NGIC
Parameter 1:      1.003656
Parameter 2:      1.001254
Parameter 3:      1.001726
Parameter 4:      1.000156
```

```
Convergence indices:TRW
Parameter 1:      1.000925
Parameter 2:      1.000707
Parameter 3:      1.001025
```

```
Convergence indices:TIC
Parameter 1:      1.000733
Parameter 2:      1.000493
Parameter 3:      1.000748
```

Some summary statistics based on the sample generated using the Normal Gamma independence chain algorithm are displayed below.

```
Summary information for posterior distribution: NGIC

Mean
Parameter 1:      0.836406
```

```
Parameter 2:      1.73749
Parameter 3:      0.0244751
Parameter 4:           1
```

Standard deviation

```
Parameter 1:      0.0413289
Parameter 2:      0.212046
Parameter 3:      0.049165
Parameter 4:      0.0708996
```

Median

```
Parameter 1:      0.832635
Parameter 2:      1.73247
Parameter 3:      0.0290645
Parameter 4:      0.997997
```

2.5 percentile

```
Parameter 1:      0.765861
Parameter 2:      1.33013
Parameter 3:     -0.0878412
Parameter 4:      0.865347
```

97.5 percentile

```
Parameter 1:      0.929479
Parameter 2:      2.16181
Parameter 3:      0.106873
Parameter 4:      1.14379
```

Histograms of the posterior samples are shown in figure 1.

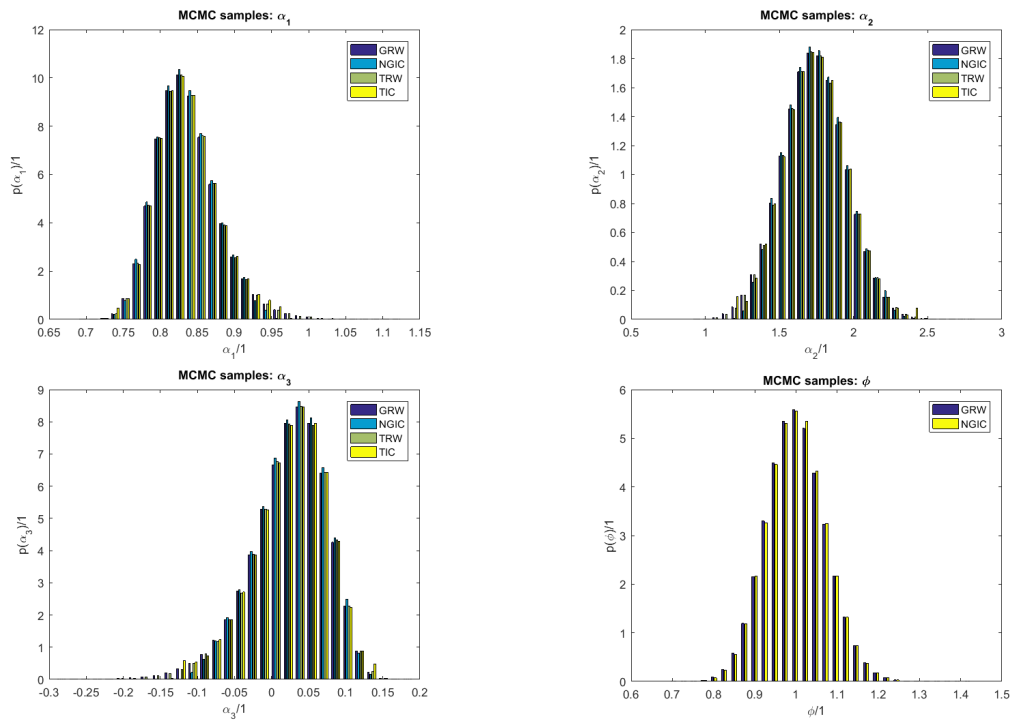


Figure 1: Posterior distributions estimated from MCMC samples for model parameters and precision parameter using various jumping distributions for the exponential decay example.

4.2 Arc fitting model

Non-linear least squares models can be used to fit geometrical elements to data. The data consists of points in a coordinate system that are assumed to have Gaussian noise. The non linear least squares fitting procedure involves minimising a function $f(\alpha, \mathbf{x})^\top f(\alpha, \mathbf{x})$ where \mathbf{x} represents points in the coordinate system. $f(\alpha, \mathbf{x})$, or f for simplicity, represents the distance of a point from the geometric element. The posterior distribution of parameters can be obtained by substituting $y - h(\alpha)$ with f in Equations (4) and (6).

The arc fitting model takes the form

$$f_i = \alpha_3 - \sqrt{(x_i - \alpha_1)^2 + (y_i - \alpha_2)^2},$$

where α_1 , α_2 and α_3 are unknown quantities.

For this problem,

$$x_i \sim N(x_i^*, \phi^{-1}\sigma_i^2), \quad y_i \sim N(y_i^*, \phi^{-1}\sigma_i^2),$$

where ϕ is an unknown precision parameter.

The posterior distributions given by Equations (4) and (6) for the arc fitting model are obtained by substituting $f_i = (\alpha_3 - \sqrt{(x_i - \alpha_1)^2 + (y_i - \alpha_2)^2})/\sigma_i$.

An illustration of the use of this software is provided using randomly simulated data along an arc of a circle. We assume parameter values $\alpha_1 = 0$, $\alpha_2 = 0$ and $\alpha_3 = 100$, and that σ_i is randomly generated between 1 and 2. An estimate $\sigma_0 = 1/\sqrt{\phi}$ is computed as the standard deviation from $m_0 = 5$ repeated measurements from a standard Gaussian distribution. Then the coordinates are derived from a Gaussian distribution with mean generated in terms of a radius of 100 at randomly generated angles and standard deviation $\sigma_0\sigma_i$.

Nonlinear least squares estimates of α are obtained using the Gauss Newton algorithm and input to the `nllsmh` modules. Since the target distribution is the same for all four methods, we expect histograms of samples from the various methods to be the same. In order to test this, the `mcmccli.m` module in the MCMCMH software package is used. This module calculates the convergence indices of MCMC chains, \hat{R} . Chains of the same parameter obtained using different methods can be compared as if they were samples from different chains. If the \hat{R} value obtained is close to one, we can conclude that the samples from the different methods do in fact arise from the same target distribution. The \hat{R} values are reported below. All the convergence indices are close to one indicating that the samples obtained are from the target distribution.

```
Rhat alpha
Parameter 1:      1.000014
Parameter 2:      1.000025
Parameter 3:      1.000014

Rhat phi:         1.000031
```


The convergence indices from each of the four algorithms is displayed below. Parameters 1-3 are the α parameters and parameter 4 is the ϕ parameter.

```
Convergence indices:GRW
Parameter 1:      1.002713
Parameter 2:      1.002174
Parameter 3:      1.002697
Parameter 4:      1.002014
```

```
Convergence indices:NGIC
Parameter 1:      1.000272
Parameter 2:      1.000085
Parameter 3:      1.000278
Parameter 4:      1.000085
```

```
Convergence indices:TRW
Parameter 1:      1.001863
Parameter 2:      1.001627
Parameter 3:      1.001875
```

```
Convergence indices:TIC
Parameter 1:      1.000390
Parameter 2:      1.000051
Parameter 3:      1.000396
```

Some summary statistics based on the sample generated using the Normal Gamma independence chain algorithm are displayed below.

Summary information for posterior distribution: NGIC

```
Mean
Parameter 1:      0.234689
Parameter 2:      -0.655717
Parameter 3:      99.7054
Parameter 4:      1.38091
```

```
Standard deviation
Parameter 1:      5.01916
Parameter 2:      0.554198
Parameter 3:      4.90163
Parameter 4:      0.193551
```

```
Median
Parameter 1:      0.475351
Parameter 2:      -0.655095
Parameter 3:      99.4619
Parameter 4:      1.37224
```

```
2.5 percentile
Parameter 1:      -10.2886
Parameter 2:      -1.74771
Parameter 3:      90.7631
Parameter 4:      1.02665
```

```
97.5 percentile
Parameter 1:      9.40418
Parameter 2:      0.434157
Parameter 3:      110.014
Parameter 4:      1.7838
```

Histograms of the posterior samples are shown in figure 2.

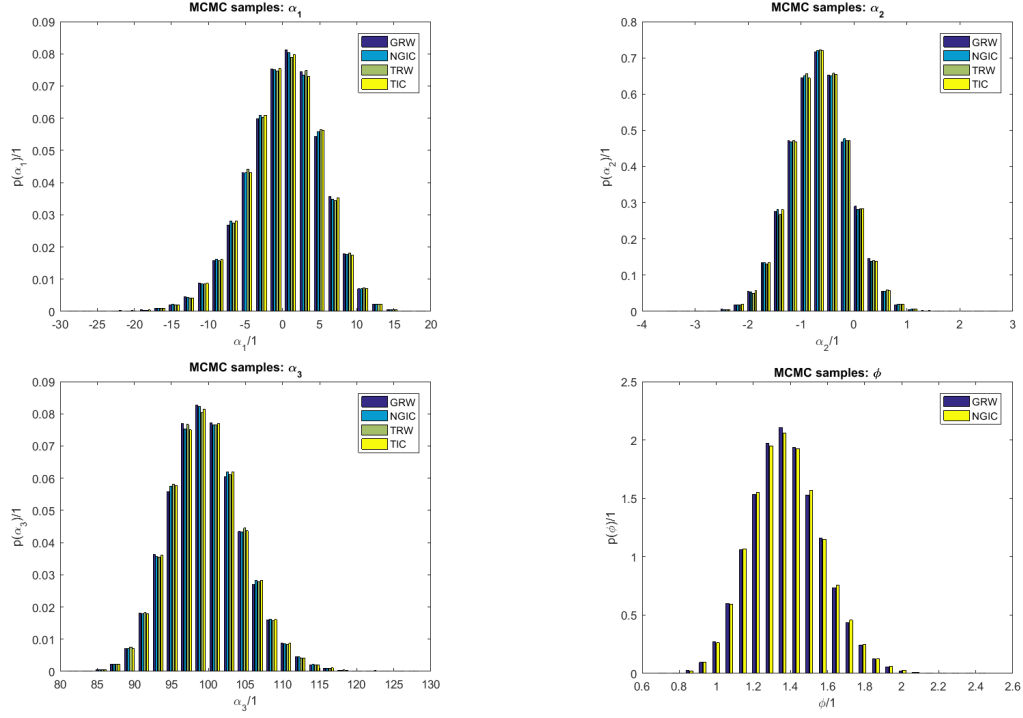


Figure 2: Posterior distributions estimated from MCMC samples for model parameters and precision parameter using various jumping distributions for the arc fitting example.

A Appendix

A.1 Change of variables formula

Let $\mathbf{x} = g(\mathbf{y})$ represent a change of variables with inverse $\mathbf{y} = f(\mathbf{x})$. If \mathbf{y} has PDF $p_Y(\mathbf{y})$ then the PDF $p_X(\mathbf{x})$ associated with \mathbf{x} is given by

$$p_X(\mathbf{x}) = |J_f(\mathbf{x})|p_Y(f(\mathbf{x})), \quad J_f(i, j) = \frac{\partial f_i}{\partial x_j},$$

if \mathbf{x} is in the range of g and is zero otherwise.

References

- [1] D. Gamerman. *Markov chain Monte Carlo: Stochastic Simulation for Bayesian inference*. Taylor & Francis, New York, 1997.
- [2] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, Fl., third edition, 2014.