# Estimating the parameters of dynamical systems from Big Data using Sequential Monte Carlo samplers

P.L.Green[a,c], S.Maskell[b,c]

[a] *School of Engineering*
[b] *Department of Electrical Engineering and Electronics*
[c] *Institute for Risk and Uncertainty*
*University of Liverpool*
*Liverpool, UK*
*L69 7ZF*

## Abstract

In this paper the authors present a method which facilitates computationally efficient parameter estimation of dynamical systems from a continuously growing set of measurement data. It is shown that the proposed method, which utilises Sequential Monte Carlo samplers, is guaranteed to be fully parallelisable (in contrast to Markov chain Monte Carlo methods) and can be applied to a wide variety of scenarios within structural dynamics. Its ability to allow *convergence* of one's parameter estimates, as more data is analysed, sets it apart from other sequential methods (such as the particle filter).

*Keywords:* Big Data, parameter estimation, model updating, system identification, Sequential Monte Carlo sampler.

## 1. Introduction

This paper addresses the situation where one is attempting to infer the parameters of a dynamical model from a large set of data which, because of its size, cannot be processed using current methods. Here, $\boldsymbol{z}_t$ denotes a vector of measurements, obtained at time $t$, and $\boldsymbol{\theta}$ is a vector of the model's parameters. The aim is to realise probabilistic estimates of $\boldsymbol{\theta}$, given the available data, via Bayes' theorem:

$$p(\boldsymbol{\theta} \,|\, \boldsymbol{z}_{1:n}) \propto p(\boldsymbol{z}_{1:n} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{1}$$

where $\boldsymbol{z}_{1:n} = \{\boldsymbol{z}_1, ..., \boldsymbol{z}_n\}$ represents the set of all measurements up to time $t = n$. In [1, 2] it was suggested that, using Markov chain Monte Carlo (MCMC) methods, one could generate samples from $p(\boldsymbol{\theta} \,|\, \boldsymbol{z}_{1:t})$ while $t$ is gradually increased. Such an approach facilitates a gradual transition from prior to posterior, which aids MCMC convergence (in a similar manner to

simulated annealing). It also allows one to analyse how one's parameter estimates converge as more data is analysed, thus helping to establish when a sufficient amount of data has been utilised. The computational cost of such an approach, however, increases dramatically as more data is analysed. This makes it poorly suited to the situation where large sets of new (potentially important) measurements are expected to arrive in the future. Other approaches such as [3] involve the selection of small subsets of 'highly informative' training data from large data sets. While this reduces computational cost, it involves the deliberate omission of measurement data which, in hindsight, may contain important information (and reduce uncertainty in the posterior as a result).

In this paper, an algorithm based on Sequential Monte Carlo (SMC) methods is proposed, which is able to address the aforementioned issues. Fundamentally, the efficiency of the method proposed here lies in its ability to exploit the inevitable redundancies that arise in large sets of measurements, as well as its suitability for modern computing architectures. It is important to note that the method proposed in this paper is different from other, recently proposed sampling methods ([4, 5] for example), as it is specifically aimed at the situation where a prohibitively large data set is available. The proposed method also allows one to track how the uncertainties in one's parameter estimates reduce as more data is analysed - thus establishing when a sufficient amount of data has been processed.

In the interest of completeness, a brief introduction to SMC methods, as well as a description of previous work relevant to the problem of interest, is given in the following section.

## 2. Sequential Monte Carlo methods

### *2.1. Importance sampling*

This section begins with a brief description of importance sampling. Here $\pi(\boldsymbol{\theta})$ is defined as a target probability distribution, from which one wishes to estimate the expected value of a function, $f(\boldsymbol{\theta})$. $\pi^*(\boldsymbol{\theta})$ is used to represent an unnormalised target, such that:

$$\pi(\boldsymbol{\theta}) = \frac{\pi^*(\boldsymbol{\theta})}{Z}, \qquad Z = \int \pi^*(\boldsymbol{\theta}) d\,\boldsymbol{\theta} \tag{2}$$

(for generality it is assumed that $Z$ is difficult to estimate here - a situation which often arises in Bayesian inference problems). The expected value of $f(\boldsymbol{\theta})$ can be written as

$$\mathrm{E}[f(\boldsymbol{\theta})] = \frac{\int f(\boldsymbol{\theta})\pi^*(\boldsymbol{\theta})d\,\boldsymbol{\theta}}{\int \pi^*(\boldsymbol{\theta})d\,\boldsymbol{\theta}} = \frac{\int f(\boldsymbol{\theta})q(\boldsymbol{\theta})w(\boldsymbol{\theta})d\,\boldsymbol{\theta}}{\int q(\boldsymbol{\theta})w(\boldsymbol{\theta})d\,\boldsymbol{\theta}} \tag{3}$$

where $w(\boldsymbol{\theta}) = \frac{\pi^*(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}$ are 'importance weights' and $q(\boldsymbol{\theta})$ is a user-defined 'proposal distribution' - a probability distribution from which it is relatively easy to generate samples. Equation (3) implies that

$$\mathrm{E}[f(\boldsymbol{\theta})] \approx \sum_{i=1}^{N} f(\boldsymbol{\theta}^i)\tilde{w}^i \tag{4}$$

where $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^N\}$ have been generated from $q(\boldsymbol{\theta})$ and, adopting the notation $w^i \equiv w(\boldsymbol{\theta}^i)$,

$$\tilde{w}^i = \frac{w^i}{\sum_j w^j}, \qquad i = 1, ..., N \tag{5}$$

are defined as 'normalised importance weights'. This reweighting procedure allows estimates of $\mathrm{E}[f(\boldsymbol{\theta})]$ to be realised using samples from $q(\boldsymbol{\theta})$, which is useful when it is difficult to generate samples from the target distribution, $\pi(\boldsymbol{\theta})$, directly.

### *2.2. Resampling*

By defining $f(\boldsymbol{\theta}_j) = \delta(\boldsymbol{\theta}_j - \boldsymbol{\theta})$ where $\delta$ is the Dirac delta function, it follows that

$$\mathrm{E}[f(\boldsymbol{\theta}_j)] = \int \delta(\boldsymbol{\theta}_j - \boldsymbol{\theta})\pi(\boldsymbol{\theta})d\,\boldsymbol{\theta} = \pi(\boldsymbol{\theta}_j). \tag{6}$$

This implies that, if one has a set of samples (and accompanying normalised weights) $\{\boldsymbol{\theta}^1, \tilde{w}^1\}, ..., \{\boldsymbol{\theta}^N, \tilde{w}^N\}$ while a new set of samples, $\{\bar{\boldsymbol{\theta}}^1, ..., \bar{\boldsymbol{\theta}}^N\}$, is chosen such that

$$\Pr(\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}^i) = \tilde{w}^i \tag{7}$$

then $\{\bar{\boldsymbol{\theta}}^1, ..., \bar{\boldsymbol{\theta}}^N\}$ will be approximate samples from the target. The weights of these new samples will therefore be equal (for more information the tutorial [6] is recommended). Resampling is often used when it is found that relatively few of the current samples have significant weight as it helps to remove those samples which are of little importance. It is often used to tackle the 'degeneracy' problem that is often encountered in the application of particle filters. To indicate when resampling is required, the concept of 'effective sample size' was introduced in [7, 8]. This involves defining

$$N_{eff} = \frac{1}{\sum_i (\tilde{w}^i)^2} \tag{8}$$

and choosing to conduct resampling when $N_{eff}$ falls below some kind of threshold ($N/2$, for example, is used throughout the current paper). It should be noted that, while resampling helps to remove 'unimportant' samples, it doesn't aid exploration of the parameter space - it can only produce replicas of the existing samples.

### *2.3. Previous work*

One of the best-known SMC methods is the particle filter, which can be used to 'track' the state of a system from a continuous stream of 'online' measurements - this method was used in [9] to monitor the time changing parameters of dynamical structures and systems. The application of a particle filter involves defining a *prediction equation*, which specifies how a system's state is expected to change (conditional on its previous state). In the scenario of interest here, justifying the choice of prediction equation would be rather difficult. Furthermore, when applying a particle filter, the influence of the initial measurements on one's parameter estimates will decrease as more data is analysed [10]. This makes particle filters poorly suited to the current application (where it is required that all available measurements, with equal weighting, are used to infer parameter estimates).

A different approach was proposed in [11] where, using the prior as a proposal distribution, importance sampling was used to target the posterior parameter distribution. At time $t$, this leads to the following expression for the importance weights:

$$w_t = \frac{p(\boldsymbol{z}_{1:t} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} = p(\boldsymbol{z}_{1:t} \,|\, \boldsymbol{\theta}). \tag{9}$$

Assuming that the probability of witnessing separate measurements is conditionally independent, such that

$$p(\boldsymbol{z}_{1:t} \,|\, \boldsymbol{\theta}) = \prod_{i=1}^{t} p(\boldsymbol{z}_i \,|\, \boldsymbol{\theta}), \tag{10}$$

equation (9) can be used to show that

$$w_t = p(\boldsymbol{z}_t \,|\, \boldsymbol{\theta}) w_{t-1}. \tag{11}$$

This allows the update of the importance weights to be conducted recursively, circumventing the need to repeatedly analyse the entire set of training data. Unfortunately, this method suffers from the same degeneracy problem as the particle filter where, after a time, only very few of the samples have significant weight. As stated previously, resampling can only help to generate replicas of these samples, and does not aid a further exploration of the parameter space. In [11] this was overcome with an 'move step' which was

facilitated using MCMC updates. This process, which involves analysis of the entire data set up to time $t$, was repeated every time a new measurement was obtained and, as a result, is computationally expensive to implement.

In this paper it is shown that these issues can be tackled efficiently and simply using a *Sequential Monte Carlo sampler*. At this point it is important to note that, somewhat confusingly, SMC *samplers* are a specific member of the family of SMC *methods* (to which particle filters also belong). A general description of SMC samplers is given in the following section. A more detailed theoretical explanation can be found in [12] while a description written within the applied context of automated navigation is given in [13].

## 3. Sequential Monte Carlo samplers

### *3.1. General formulation*

Here, $\boldsymbol{\theta}_k$ is used to represent the state of a system at iteration $k$, $\pi_k(\boldsymbol{\theta}_k)$ is defined as the $k$th target distribution and $\pi(\boldsymbol{\theta}_{1:k})$ represents the joint distribution over $\boldsymbol{\theta}_{1:k}$. One begins by defining

$$\pi(\boldsymbol{\theta}_{1:k}) = \pi_k(\boldsymbol{\theta}_k) \prod_{k'=2}^{k} L(\boldsymbol{\theta}_{k'-1} \,|\, \boldsymbol{\theta}_{k'}) \tag{12}$$

where $L(\boldsymbol{\theta}_{k'-1} \,|\, \boldsymbol{\theta}_{k'})$ - the 'L-kernel' - is a design parameter which is defined such that

$$\int \pi(\boldsymbol{\theta}_{1:k}) d\,\boldsymbol{\theta}_{1:k-1} = \pi_k(\boldsymbol{\theta}_k) \tag{13}$$

(thus realising the property that $\int f(\boldsymbol{\theta}_k)\pi(\boldsymbol{\theta}_{1:k}) d\,\boldsymbol{\theta}_{1:k} = \int f(\boldsymbol{\theta}_k)\pi_k(\boldsymbol{\theta}_k) d\,\boldsymbol{\theta}_k$). Potential choices for the L-kernel are described subsequently. With proposal distribution $q(\boldsymbol{\theta}_{1:k})$, the importance weights are defined as

$$w_k^i = \frac{\pi(\boldsymbol{\theta}_{1:k}^i)}{q(\boldsymbol{\theta}_{1:k}^i)}, \qquad \boldsymbol{\theta}_{1:k}^i \sim q(\boldsymbol{\theta}_{1:k}). \tag{14}$$

Choosing a proposal of the form $q(\boldsymbol{\theta}_{1:k}) = q(\boldsymbol{\theta}_k \,|\, \boldsymbol{\theta}_{k-1}) q(\boldsymbol{\theta}_{1:k-1})$ then allows one to write

$$w_k^i = \frac{\pi_k(\boldsymbol{\theta}_k^i) \prod_{k'=2}^{k} L(\boldsymbol{\theta}_{k'-1}^i \,|\, \boldsymbol{\theta}_{k'}^i)}{q(\boldsymbol{\theta}_k^i \,|\, \boldsymbol{\theta}_{k-1}^i) q(\boldsymbol{\theta}_{1:k-1}^i)}. \tag{15}$$

Dividing $w_k^i$ by $w_{k-1}^i$, it is then possible to show that

$$w_k^i = w_{k-1}^i \frac{\pi_k(\boldsymbol{\theta}_k^i)}{\pi_{k-1}(\boldsymbol{\theta}_{k-1}^i)} \frac{L(\boldsymbol{\theta}_{k-1}^i \,|\, \boldsymbol{\theta}_k^i)}{q(\boldsymbol{\theta}_k^i \,|\, \boldsymbol{\theta}_{k-1}^i)} \tag{16}$$

such that the importance weights can be sequentially updated as $k$ increases. At first glance one may consider replacing the index $k$ with the index $t$ and applying the SMC sampler directly to the problem described in Section 1, such that

$$\pi_t(\boldsymbol{\theta}_t) \equiv p(\boldsymbol{\theta} \,|\, \boldsymbol{z}_{1:t}). \tag{17}$$

This will, however, involve analysis of the full data set every time new measurements are analysed. The computational cost of this would make such an approach impractical. In the current work, SMC samplers are actually used to facilitate the *resampling* step of the method proposed in [11] (hence the disparity between the indexes $k$ and $t$). Before this process can be described fully, it is worth stating how SMC samplers can be used to sample from a stationary target - one which doesn't vary with the index $k$.

### *3.2. Sampling from an invariant target*

Consider the situation where one wishes to estimate the mean of an (unnormalised) target distribution $\pi^*(\boldsymbol{\theta})$. Here, the index $k$ is simply used to denote the $k$th estimate of the mean. Having defined the proposal distributions $q(\boldsymbol{\theta}_1)$ and $q(\boldsymbol{\theta}_k \,|\, \boldsymbol{\theta}_{k-1})$, algorithm 1 shows how this can be achieved using a SMC sampler.

**Algorithm 1** Targeting a stationary distribution using a SMC sampler.

$k = 1$
Sample $\{\boldsymbol{\theta}_k^1, ..., \boldsymbol{\theta}_k^N\}$ from $q(\boldsymbol{\theta}_k)$
Initial weights: $w_k^i = \frac{\pi^*(\boldsymbol{\theta}_k^i)}{q(\boldsymbol{\theta}_k^i)}, \qquad i = 1, ..., N$
**while do**
  Normalise weights: $\tilde{w}^i = \frac{w_k^i}{\sum_j w_k^j}, \qquad i = 1, ..., N$
  Estimate quantities of interest
  $N_{eff} = \frac{1}{\sum_i (\tilde{w}^i)^2}$
  **if** $N_{eff} < N/2$ **then**
    Resample to get $\{\boldsymbol{\theta}_k^1, ..., \boldsymbol{\theta}_k^N\}$
    Reset weights: $w_k^i = 1, \qquad i = 1, ..., N$
  **end if**
  $k = k + 1$
  Sample $\{\boldsymbol{\theta}_k^1, ..., \boldsymbol{\theta}_k^N\}$ from $q(\boldsymbol{\theta}_k \,|\, \boldsymbol{\theta}_{k-1}^i)$
  New weights: $w_k^i = w_{k-1}^i \frac{\pi^*(\boldsymbol{\theta}_k^i)}{\pi^*(\boldsymbol{\theta}_{k-1}^i)} \frac{L(\boldsymbol{\theta}_{k-1}^i \,|\, \boldsymbol{\theta}_k^i)}{q(\boldsymbol{\theta}_k^i \,|\, \boldsymbol{\theta}_{k-1}^i)}, \qquad i = 1, ..., N$
**end while**

Note that choosing the L-kernel

$$L(\boldsymbol{\theta}_{k-1} \,|\, \boldsymbol{\theta}_k) = q(\boldsymbol{\theta}_{k-1} \,|\, \boldsymbol{\theta}_k) \tag{18}$$

allows the weights to be updated according to

$$w_k^i = w_{k-1}^i \frac{\pi^*(\boldsymbol{\theta}_k^i)}{\pi^*(\boldsymbol{\theta}_{k-1}^i)}. \tag{19}$$

As an example, a SMC sampler will be used to estimate the mean of the target distribution

$$\pi(\theta) = \mathcal{N}(\theta; \mu, \sigma^2) \tag{20}$$

where $\mu = 5$ and $\sigma = 0.5$. For this example the proposal distributions and L-kernel are defined as:

$$q(\theta_1) = \mathcal{N}(\theta_1; 0, 1), \quad q(\theta_k|\theta_{k-1}) = \mathcal{N}(\theta_k; \theta_{k-1}, 1),$$

$$L(\theta_{k-1}|\theta_k) = \mathcal{N}(\theta_{k-1}; \theta_k, 1). \tag{21}$$

Using 100 samples, the resulting estimate of the mean, as a function of $k$, is shown in Figure 1.

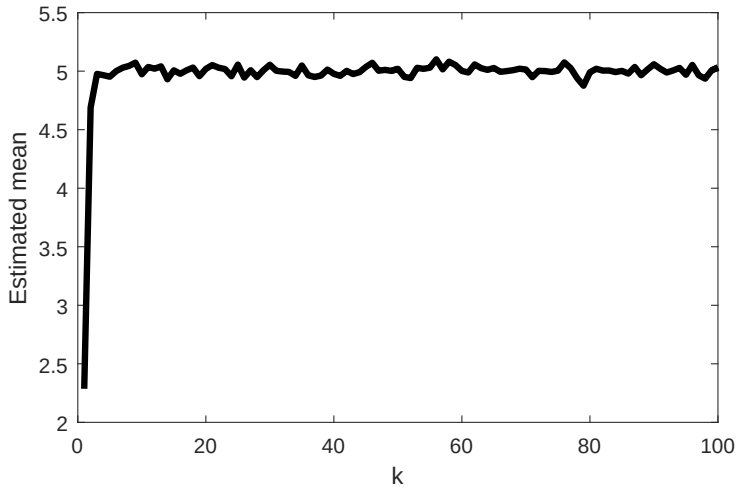


Figure 1: Estimating the mean of a stationary distribution using a SMC sampler.

In the formulation shown in algorithm 1, all previous samples of $\boldsymbol{\theta}$ are 'saved' as $k$ increases. In the following applications this isn't necessary - algorithm 2 therefore shows an alternative form of algorithm 1, where this notation has been dropped (this also establishes a clearer notation for the

algorithms described later in the paper).

**Algorithm 2** Targeting a stationary distribution using a SMC sampler (without the index $k$).

Sample $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^N\}$ from $q(\boldsymbol{\theta}_1)$
Initial weights: $w^i = \frac{\pi^*(\boldsymbol{\theta}^i)}{q(\boldsymbol{\theta}^i)}, \qquad i = 1, ..., N$
**while do**
  Normalise weights: $\tilde{w}^i = \frac{w^i}{\sum_j w^j}, \qquad i = 1, ..., N$
  Estimate quantities of interest
  $N_{eff} = \frac{1}{\sum_i (\tilde{w}^i)^2}$
  **if** $N_{eff} < N/2$ **then**
    Resample to get $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^N\}$
    Reset weights: $w^i = 1, \qquad i = 1, ..., N$
  **end if**
  Sample $\{\hat{\boldsymbol{\theta}^1}, ..., \hat{\boldsymbol{\theta}^N}\}$ from $q(\hat{\boldsymbol{\theta}} | \, \boldsymbol{\theta}^i)$
  New weights: $\hat{w}^i = w^i \frac{\pi^*(\hat{\boldsymbol{\theta}}^i)}{\pi^*(\boldsymbol{\theta}^i)} \frac{L(\boldsymbol{\theta}^i \, | \hat{\boldsymbol{\theta}}^i)}{q(\hat{\boldsymbol{\theta}}^i | \, \boldsymbol{\theta}^i)}, \qquad i = 1, ..., N$
  Set $w^i = \hat{w}^i, \boldsymbol{\theta}^i = \hat{\boldsymbol{\theta}}^i, \qquad i = 1, ..., N$
**end while**

## 4. Parameter estimation of static models

In this section it will be shown how, by using a SMC sampler to facilitate *resampling*, it is possible to conduct the sequential parameter estimation of static models from large sets of data. (Here the phrase 'static models' is used to distinguish the current example from the dynamical, autoregressive models investigated later in the paper).

### *4.1. Algorithm*

Consider a model of the form

$$\boldsymbol{y}_t = f(\boldsymbol{x}_t, \boldsymbol{\theta}) \tag{22}$$

where $\boldsymbol{x}_t$ represents an input to the system (at time $t$) and, as before, $\boldsymbol{\theta}$ is a vector of the model's parameters. It is assumed that measurements are realised according to

$$\boldsymbol{z}_t = h(\boldsymbol{y}_t) + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \boldsymbol{\Sigma}_\epsilon) \tag{23}$$

(such that $\boldsymbol{\epsilon}$ represents measurement noise) where the covariance matrix, $\boldsymbol{\Sigma}_\epsilon$, is assumed to be known. Finally, it is also assumed that the probability of witnessing sequential measurements are conditionally independent (as in equation (10)). Algorithm 3 shows how, using the proposed method,

sequential estimates of the model's parameters can be realised. The method proceeds in a similar way to that proposed in [11], until the effective sample size drops below a predefined threshold. When this occurs, resampling is used to remove those samples with low weights, before a SMC sampler is used to facilitate 'movement' - to explore the parameter space. It is important to note that the SMC sampler must be allowed to run *until* the effective sample size has recovered and that the quantities of interest are always estimated *after* the SMC sampler has run (this prevents them from being based on samples with a low effective sample size). While the SMC sampling step will involve a 'full analysis' of all the existing data, algorithm 3 has the following useful properties:

1. The SMC sampler will only be used if the effective sample size drops below a predefined threshold.
2. The SMC sampler is well suited to parallel processing, thus allowing the full exploitation of modern computing architectures.

**Algorithm 3** Sequential parameter estimation of a static model using the proposed methodology.

Sample $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^N\}$ from $p(\boldsymbol{\theta})$
Initial weights: $w_1^i = p(\boldsymbol{z}_1 \,|\, \boldsymbol{\theta}^i), \qquad i = 1, ..., N$
$t = 1$
**while do**
  Normalise weights: $\tilde{w}^i = \frac{w_t^i}{\sum_j w_t^j}, \qquad i = 1, ..., N$
  $N_{eff} = \frac{1}{\sum_i (\tilde{w}^i)^2}$
  **while** $N_{eff} < N/2$ **do**
    Resample to get $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^N\}$
    Reset weights: $w_t^i = 1, \qquad i = 1, ..., N$
    Sample $\{\hat{\boldsymbol{\theta}}^1, ..., \hat{\boldsymbol{\theta}}^N\}$ from $q(\hat{\boldsymbol{\theta}} | \, \boldsymbol{\theta}^i)$
    New weights: $\hat{w}^i = w_t^i \frac{p(\boldsymbol{z}_{1:t} \,|\, \hat{\boldsymbol{\theta}}^i)}{p(\boldsymbol{z}_{1:t} \,|\, \boldsymbol{\theta}^i)} \frac{L(\boldsymbol{\theta}^i \,|\, \hat{\boldsymbol{\theta}}^i)}{q(\hat{\boldsymbol{\theta}}^i \,|\, \boldsymbol{\theta}^i)}, \qquad i = 1, ..., N$
    Set $w_t^i = \hat{w}^i$, $\boldsymbol{\theta}^i = \hat{\boldsymbol{\theta}}^i$ and $y_{1:t}^i = \hat{y}_{1:t}^i, \qquad i = 1, ..., N$
    Normalise weights: $\tilde{w}^i = \frac{w_t^i}{\sum_j w_t^j}$
    $N_{eff} = \frac{1}{\sum_i (\tilde{w}^i)^2}$
  **end while**
  Estimate quantities of interest
  $t = t + 1$
  New weights: $w_t^i = w_{t-1}^i p(\boldsymbol{z}_t \,|\, \boldsymbol{\theta}^i)$
**end while**

*4.2. Choice of proposal density*

The efficiency and repeatability of any algorithm which utilises importance sampling is heavily dependent on the choice of proposal distribution. Consider, again, the situation where the aim is to estimate the quantity $\mathrm{E}[f(\boldsymbol{\theta})] = \int f(\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}$ using importance sampling. Writing the resulting estimate as

$$\hat{f} = \sum_{i=1}^{N} f(\boldsymbol{\theta}^i)\tilde{w}^i \tag{24}$$

(where $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^N\}$ have been generated from the proposal distribution $q(\boldsymbol{\theta})$), then it is possible to show that $\hat{f}$ is an unbiased estimator, such that

$$\mathrm{E}[\hat{f}] = \mathrm{E}[f(\boldsymbol{\theta})]. \tag{25}$$

The variance of $\hat{f}$ can also be shown to be

$$\mathrm{Var}[\hat{f}] = \frac{c}{N}\,\mathrm{E}[f^2(\boldsymbol{\theta})] \tag{26}$$

where $c$ is a constant and it has been assumed that

$$\frac{\pi(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} < c \tag{27}$$

for all values of $\boldsymbol{\theta}$ [10]. Equations (26) and (27) illustrate that, to prevent $\mathrm{Var}[\hat{f}]$ from being large, one must choose a proposal distribution that is *heavier-tailed* than the target. This fact is not unique to SMC samplers - it is well established that the use of heavy-tailed proposal distributions in the application of MCMC methods can aid performance (see [1], for example, where a Cauchy distribution was used to aid MCMC convergence). To avoid large numbers of 'wasted samples', however, the proposal should also be designed such that majority of the samples generated will fall in the area of interest.

In the examples described here, Gaussian proposals, scaled relative to the variance of the target distribution, were utilised. Specifically, defining $\boldsymbol{\Sigma}_t$ as the current estimate of the target distribution's covariance matrix, proposals were made according to

$$q(\hat{\boldsymbol{\theta}}|\,\boldsymbol{\theta}^i) = \mathcal{N}(\hat{\boldsymbol{\theta}}; \boldsymbol{\theta}^i, 0.1\boldsymbol{\Sigma}_t) \qquad \text{and} \qquad q(\hat{\boldsymbol{\theta}}|\,\boldsymbol{\theta}^i) = \mathcal{N}(\hat{\boldsymbol{\theta}}; \boldsymbol{\theta}^i, \boldsymbol{\Sigma}_t) \tag{28}$$

with probabilities 0.9 and 0.1 respectively. Such a strategy is often referred to as 'defensive sampling' [14] and increases the repeatability of the SMC sampler. It should also be observed that, for all the examples in this paper, the L-kernel was chosen such that $L(\boldsymbol{\theta}\,|\hat{\boldsymbol{\theta}})/q(\hat{\boldsymbol{\theta}}|\,\boldsymbol{\theta}) = 1$.

*4.3. Example - linear static system*

As an example, a linear model will be considered:

$$y_t = \theta x_t, \tag{29}$$

where measurements are made according to

$$z_t = y_t + \epsilon, \qquad \epsilon \sim \mathcal{N}(\epsilon; 0, \beta_\epsilon^{-1}), \tag{30}$$

$x$ is the system's input and $\epsilon$ is noise with precision $\beta_\epsilon$ (which is assumed known). After obtaining $t$ measurements, the likelihood is

$$p(\boldsymbol{z}_{1:t}\,|\theta) \propto \exp\left(-\frac{\beta_\epsilon}{2}\sum_{t'=1}^{t}(z_{t'} - \theta x_{t'})^2\right). \tag{31}$$

Choosing a Gaussian prior:

$$p(\theta) \propto \exp\left(-\frac{\beta_0}{2}(\theta - \mu_0)^2\right), \tag{32}$$

it can be shown that the posterior distribution is

$$p(\theta|\,\boldsymbol{z}_{1:t}) \propto \exp\left(-\frac{\beta}{2}(\theta - \mu)^2\right) \tag{33}$$

where

$$\beta = \beta_\epsilon \sum_{t'=1}^{t} x_{t'}^2 + \beta_0, \qquad \mu = \frac{1}{\beta}\left(\beta_\epsilon \sum_{t'=1}^{t} x_{t'} z_{t'} + \beta_0 \mu_0\right). \tag{34}$$

The goal here is to track the mean and variance of $\theta$ as $t$ increases. This example was chosen because, as an analytical expression for the posterior is available, it allows one to verify the proposed sampling algorithm. To create a set of training data, samples of $x_t$ were generated from a uniform distribution (between 0 and 1) and an artificial set of 'measurements' were created. The prior moments and noise precision were:

$$\mu_0 = 0, \quad \beta_0 = 1, \quad \beta_\epsilon = 100. \tag{35}$$

A sample size of 100 was used throughout this example.

Figure 2 shows that the estimates realised using the proposed sampling method closely match the true solution. The red circles in Figure 2 indicate where resampling occurred. It can be observed that, as the parameter estimate converges, resampling is needed less frequently (as additional data is only leading to very small changes in the geometry of the target distribution).

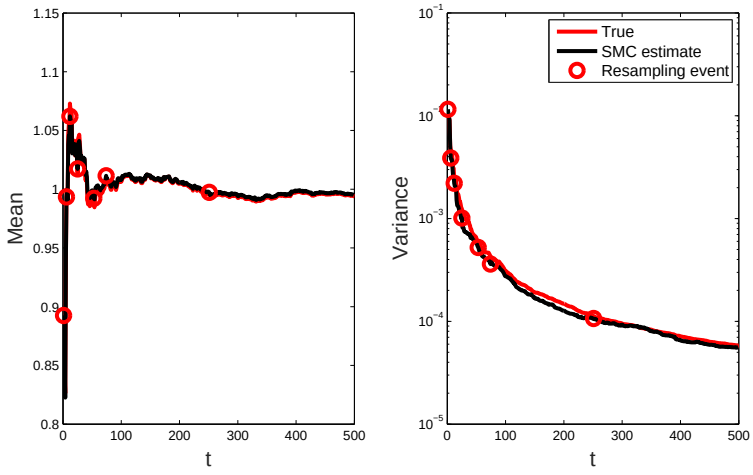


Figure 2: Estimating the mean (a) and variance (b) of $\theta$ (the parameter of the model shown in equation (29)). Red lines indicate the true solution, black lines indicate results according to the proposed sampling method and red circles indicate where resampling occurred.

## 5. Parameter estimation of dynamical models

### *5.1. Algorithm*

Consider a dynamical model, which makes predictions according to

$$\boldsymbol{y}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}) \tag{36}$$

where $\boldsymbol{\theta}$ are the model's parameters and $\boldsymbol{x}_t$ is the input to the system. Defining $\boldsymbol{x}_0$ and $\boldsymbol{y}_0$ as initial conditions (which are assumed known) then, in this notation, it follows that

$$\boldsymbol{y}_{1:t} = f(\boldsymbol{y}_0, \boldsymbol{x}_0, \boldsymbol{x}_{1:t}, \boldsymbol{\theta}). \tag{37}$$

Again, measurements are related to the state of the system by

$$\boldsymbol{z}_t = h(\boldsymbol{y}_t) + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \boldsymbol{\Sigma}_\epsilon). \tag{38}$$

The likelihood of the parameters $\boldsymbol{\theta}$, given measurement $\boldsymbol{z}_t$, is therefore:

$$p(\boldsymbol{z}_t \,|\, \boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{z}_t; h(\boldsymbol{y}_t), \boldsymbol{\Sigma}_\epsilon\right) \tag{39}$$

where the independence property (equation (10)) has, once again, been assumed. Algorithm 4 shows how, using the proposed methodology, it is possible to estimate the parameters of a dynamical model from a continuous stream of measurement data.

**Algorithm 4** Sequential parameter estimation of a dynamical model using the proposed methodology.

Set initial conditions ($\boldsymbol{x}_0$ and $\boldsymbol{y}_0$)
Sample $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^N\}$ from $p(\boldsymbol{\theta})$
Find $\boldsymbol{y}_1^i = f(\boldsymbol{y}_0, \boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{\theta}^i), \qquad i = 1, ..., N$
Initial weights: $w_1^i = p(\boldsymbol{z}_1 \,|\, \boldsymbol{\theta}^i), \qquad i = 1, ..., N$
$t = 1$
**while** **do**
  Normalise weights: $\tilde{w}^i = \frac{w_t^i}{\sum_j w_t^j}, \qquad i = 1, ..., N$
  $N_{eff} = \frac{1}{\sum_i (\tilde{w}^i)^2}$
  **while** $N_{eff} < N/2$ **do**
    Resample to get $\{\boldsymbol{\theta}^1, ..., \boldsymbol{\theta}^N\}$
    Reset weights: $w_t^i = 1, \qquad i = 1, ..., N$
    Sample $\{\hat{\boldsymbol{\theta}}^1, ..., \hat{\boldsymbol{\theta}}^N\}$ from $q(\hat{\boldsymbol{\theta}} | \, \boldsymbol{\theta}^i)$
    Find $\hat{y}_{1:t}^i = f(\boldsymbol{y}_0, \boldsymbol{x}_0, \boldsymbol{x}_{1:t}, \hat{\boldsymbol{\theta}}), \qquad i = 1, ..., N$
    New weights: $\hat{w}^i = w_t^i \frac{p(\boldsymbol{z}_{1:t} \,|\, \hat{\boldsymbol{\theta}}^i)}{p(\boldsymbol{z}_{1:t} \,|\, \boldsymbol{\theta}^i)} \frac{L(\boldsymbol{\theta}^i \,|\, \hat{\boldsymbol{\theta}}^i)}{q(\hat{\boldsymbol{\theta}}^i | \, \boldsymbol{\theta}^i)}, \qquad i = 1, ..., N$
    Set $w_t^i = \hat{w}^i$, $\boldsymbol{\theta}^i = \hat{\boldsymbol{\theta}}^i$ and $y_{1:t}^i = \hat{y}_{1:t}^i, \qquad i = 1, ..., N$
    Normalise weights: $\tilde{w}^i = \frac{w_t^i}{\sum_j w_t^j}$
    $N_{eff} = \frac{1}{\sum_i (\tilde{w}^i)^2}$
  **end while**
  Estimate quantities of interest
  $t = t + 1$
  Find $\boldsymbol{y}_t^i = f(\boldsymbol{y}_{t-1}^i, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}^i), \qquad i = 1, ..., N$
  New weights: $w_t^i = w_{t-1}^i p(\boldsymbol{z}_t \,|\, \boldsymbol{\theta}^i), \qquad i = 1, ..., N$
**end while**

*5.2. Example - linear dynamical system*

In this section the proposed methodology is used to identify (and track) the parameters of a shear model of a two storey structure, subject to a base acceleration, $\ddot{x}$. This initial case study is conducted using simulated training data. The vector $\boldsymbol{y}$ is defined such that

$$\boldsymbol{y}_t \equiv \left\{ \begin{array}{c} y_1(t) \\ y_2(t) \end{array} \right\} \tag{40}$$

where $y_1(t)$ is the relative displacement between the ground and floor 1 while $y_2(t)$ is the relative displacement between the ground and floor 2 (at time $t$). The structure's equation of motion is

$$\boldsymbol{M}\,\ddot{\boldsymbol{y}} + \boldsymbol{C}\,\dot{\boldsymbol{y}} + \boldsymbol{K}\,\boldsymbol{y} = \boldsymbol{M}\,\ddot{\boldsymbol{x}} \tag{41}$$

where $\boldsymbol{M} = \text{diag}(m_1, m_2)$ and

$$\boldsymbol{K} = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix}, \quad \ddot{\boldsymbol{x}} = \begin{pmatrix} \ddot{x} \\ \ddot{x} \end{pmatrix}. \quad (42)$$

$m$'s represent the masses of the two floors, $k$'s represent the magnitudes of the inter-storey stiffnesses and $c$'s represent the magnitude of the inter-storey damping. It was assumed that the masses of the floors were known, so the set of parameters to be estimated was $\boldsymbol{\theta} = \{k_1, k_2, c_1, c_2\}$. The masses of the two floors were set equal to 10000 kg while the values of the other model parameters were set as shown in Table 1. The hypothetical structure had natural frequencies equal to 5 and 12.3 Hz, while the inter-storey modal damping ratios were both between 1 and 5 %. Details of the priors used for estimation of the parameters are given in Table 1. The base acceleration of the structure was sampled from a Gaussian distribution while, to create a synthetic set of 'measurements', the resulting displacement time histories of the two stories were corrupted with Gaussian noise. As stated previously, throughout the following examples, $L(\boldsymbol{\theta}\,|\hat{\boldsymbol{\theta}})$ was set equal to $q(\hat{\boldsymbol{\theta}}|\,\boldsymbol{\theta})$.

| Parameter | True value | Prior |
|---|---|---|
| $k_1$ | $30 \times 10^6$ | $\mathcal{N}(k_1; 25 \times 10^6, 2.5 \times 10^{12})$ |
| $k_2$ | $20 \times 10^6$ | $\mathcal{N}(k_2; 25 \times 10^6, 2.5 \times 10^{12})$ |
| $c_1$ | $30 \times 10^3$ | $\Gamma(c_1; 1, 5 \times 10^4)$ |
| $c_2$ | $20 \times 10^3$ | $\Gamma(c_2; 1, 5 \times 10^4)$ |

Table 1: True parameter values and their corresponding priors (all SI units).

The algorithm was applied to a scenario where $\mathrm{Std}[\ddot{x}] = 0.01\mathrm{m/s}^2$. Because of the relatively low excitation amplitude, the resulting measurements were relatively close to the noise floor (the signal to noise ratio was approximately equal to 7). Figure 3 shows how the parameter estimates converge as the first 2000 data points are analysed while Figure 4 shows that, as more data is analysed, resampling in needed less frequently. This indicates that, as a larger set of data is analysed, less new information is obtained. Figures 3 and 4 also illustrate how the proposed method can help to establish when a sufficient amount of training data has been analysed. Figure 5 shows histograms of the samples when $t = 2000$.

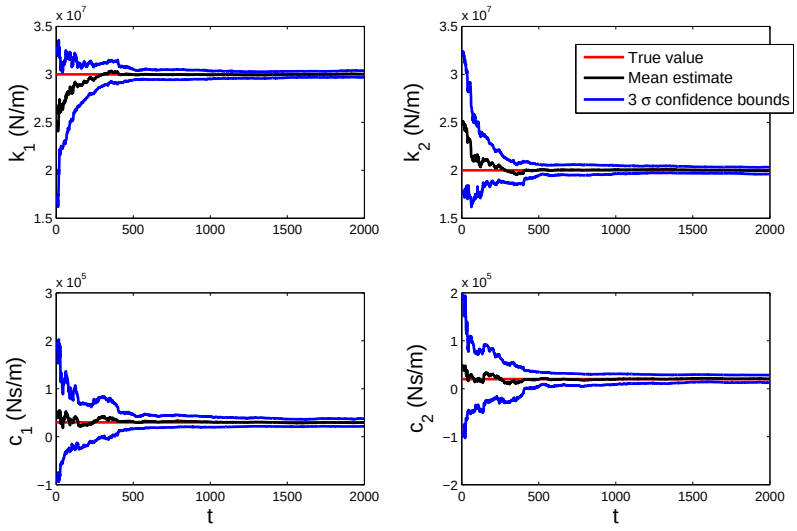


Figure 3: Identifying the parameters of the linear shear building. Convergence of the parameter estimates towards their true value.

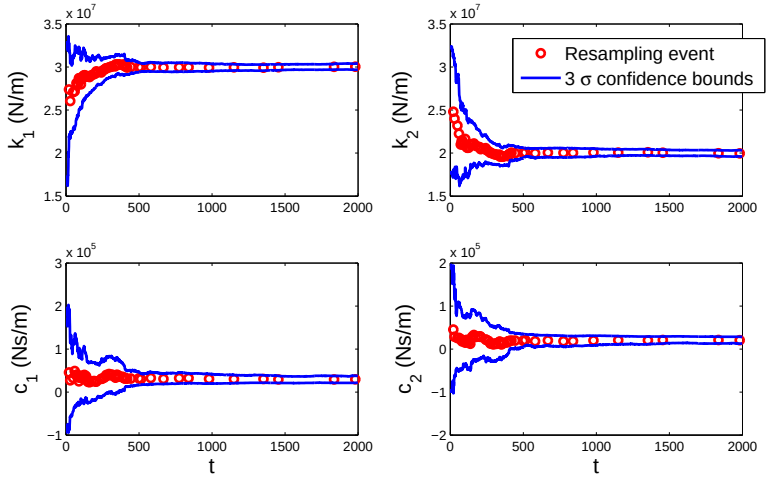


Figure 4: Identifying the parameters of the linear shear building. Resampling events required during convergence.

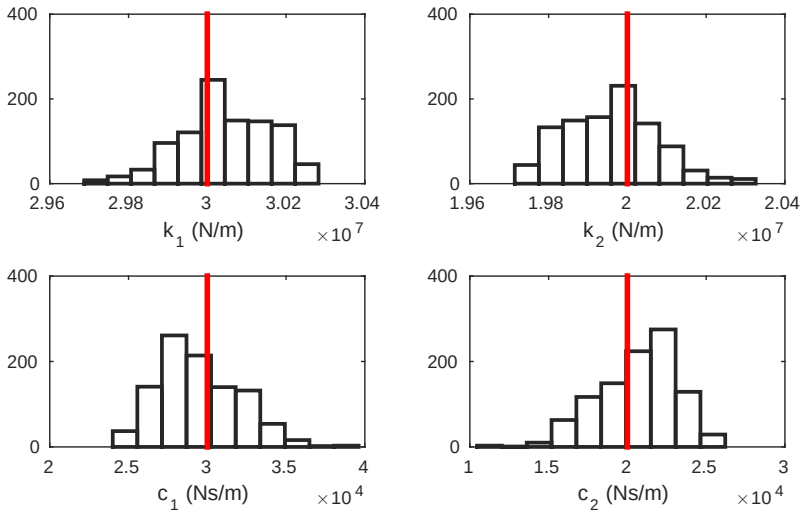


Figure 5: Identifying the parameters of the linear shear building. Histogram of samples taken when $t = 2000$. Red lines represent true parameter values.

For further validation, the TMCMC algorithm [15] was used to generate posterior samples for the cases where $t = 50, 100, 150$ and 200. Each run of the TMCMC algorithm was repeated 10 times to account for the inherent variability in results. Figure 6 shows that there is a close agreement between the results realised using TMCMC and the proposed method. It also highlights how, to track how the parameter estimates converge as more data is analysed, the TMCMC algorithm has to be run repeatedly for each different value of $t$ that is of interest. This is very computationally expensive relative to the proposed method, where such repeated runs are not needed. As a final piece of validation, Figure 7 illustrates that the empirical cumulative distribution functions realised using the proposed method, when $t = 200$, are nested within the ensemble of 10 results that were realised using TMCMC.

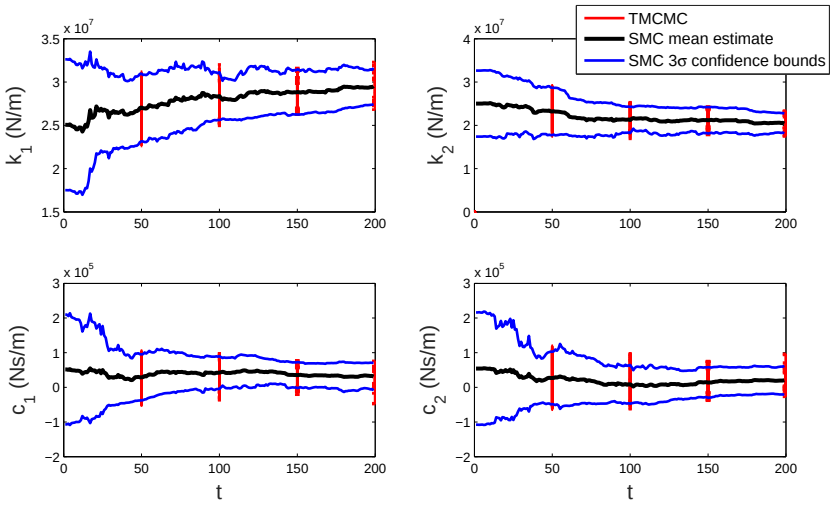


Figure 6: Identifying the parameters of the linear shear building - validating the proposed approach using results from the TMCMC [15] algorithm. Error bars on the results realised using TMCMC represent $3\sigma$ confidence bounds.

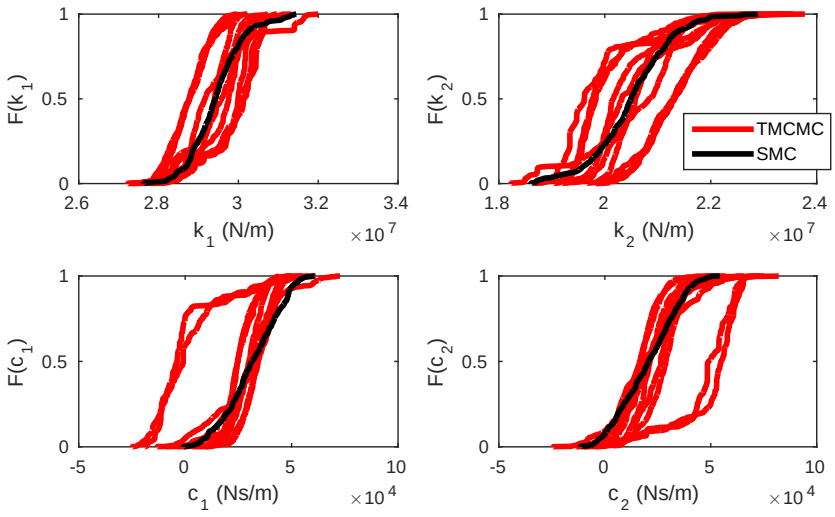


Figure 7: Identifying the parameters of the linear shear building - empirical cumulative distribution functions realised when $t = 200$.

At this point it should be observed that data does not necessarily have to be introduced to algorithm 4 a single point at a time - some simple alterations will also allow it to process data which arrives in separate 'blocks' (from separate experiments, for example). Such a situation would lead to one targeting the sequence of distributions

$$p(\boldsymbol{\theta} \,|\, \boldsymbol{z}_{1:t\Delta T}), \quad t = 1, 2, 3, ... \tag{43}$$

where $\Delta T$ is an integer that is larger than unity[1]. To accommodate this, the final line of algorithm 4 (where the 'new weights' are updated), simply needs to be changed to

$$w_t = p(\boldsymbol{z}_{(t-1)\Delta T+1:t\Delta T} \,|\, \boldsymbol{\theta}) w_{t-1}. \tag{44}$$

To demonstrate this, the data generated from the linear shear building model was reanalysed, with $\Delta T$ set equal to 50. The results shown in Figure 8 were realised.

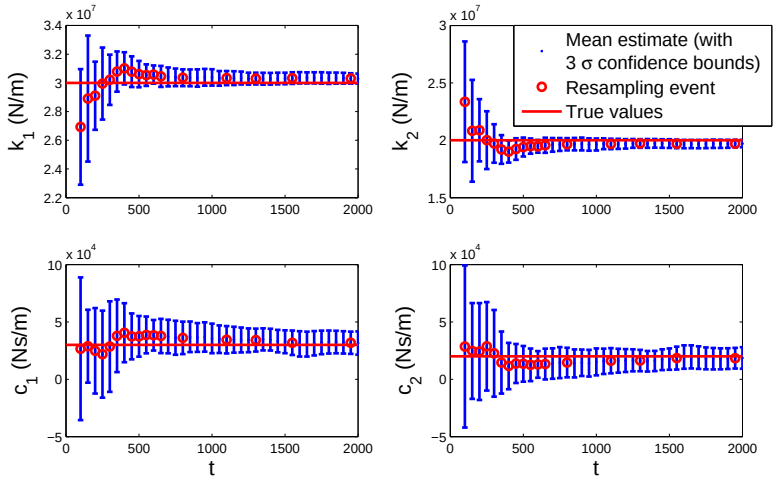


Figure 8: Identifying the parameters of the linear shear building, where data is delivered in 'blocks' of 50 points.

### *5.3. Example - nonlinear dynamical system*

In this section, the aforementioned structure was modified, such that it had two cubic, softening nonlinear stiffness terms between the two floors. The values of the nonlinear stiffnesses were chosen such that, under a relatively large base acceleration ($\text{Std}[\ddot{x}] = 0.1\text{m/s}^2$), a drop in the structure's natural frequencies would occur - a feature which can sometimes be observed in tall buildings [16]. The system's equation of motion was therefore

$$\boldsymbol{M}\,\ddot{\boldsymbol{y}} + \boldsymbol{C}\,\dot{\boldsymbol{y}} + \boldsymbol{K}\,\boldsymbol{y} + \boldsymbol{\eta}(\boldsymbol{y}) = \boldsymbol{M}\,\ddot{\boldsymbol{x}} \tag{45}$$

where

[1]Implementing this approach in the scenario where $\Delta T$ is allowed to vary can also be accomplished relatively easily. A constant $\Delta T$ is investigated here as it helps to maintain notational simplicity.

$$\boldsymbol{\eta} = \begin{pmatrix} -k_1^* y_1^3 + k_2^*(y_2 - y_1)^3 \\ -k_2^*(y_2 - y_1)^3 \end{pmatrix} \tag{46}$$

(such that $k_1^*$ and $k_2^*$ represented the nonlinear stiffness terms). The set of parameters to be identified was therefore $\boldsymbol{\theta} = \{k_1, k_2, c_1, c_2, k_1^*, k_2^*\}$. The true values and prior distributions of $k_1^*$ and $k_2^*$ are shown in Table 2 (all other values are as shown in Table 1).

| Parameter | True value | Prior |
|---|---|---|
| $k_1^*$ | $1 \times 10^{13}$ | $\mathcal{U}(k_1^*; 0, 1 \times 10^{15})$ |
| $k_2^*$ | $1 \times 10^{13}$ | $\mathcal{U}(k_2^*; 0, 1 \times 10^{15})$ |

Table 2: True nonlinear stiffness parameter values and their corresponding priors (all SI units).

A set of training data was created using a low amplitude excitation ($\text{Std}[\ddot{x}] = 0.01\text{m/s}^2$) except for a high amplitude 'event' where, between $t = 500$ and $t = 1000$, the system experiences a much larger base acceleration ($\text{Std}[\ddot{x}] = 0.1\text{m/s}^2$). The resulting training data is shown in Figure 9. Here then, the ability of the algorithm to process data which features a sudden influx of information is being analysed.

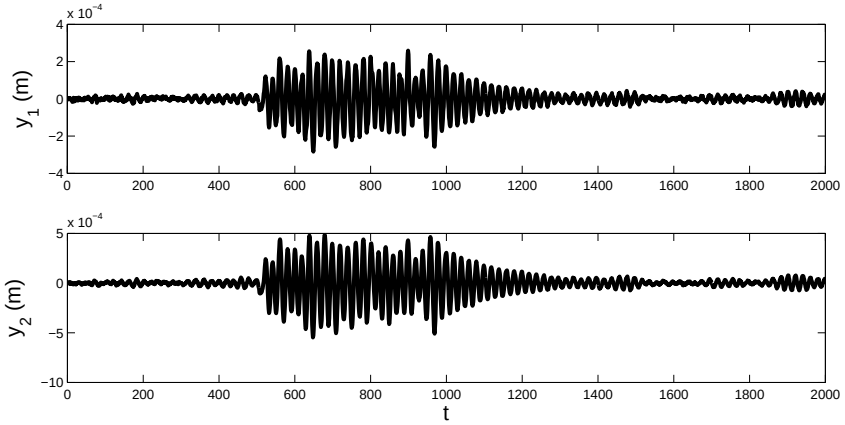


Figure 9: Training data created using the nonlinear model, equation (45). A high amplitude 'event' has deliberately been inserted between $t = 500$ and $t = 1000$.

Figure 10 shows how, as one would expect, there is clearly very little information in the low amplitude data which would allow estimation of the nonlinear stiffness terms. Furthermore, the algorithm is able to quickly respond to the sudden influx of data introduced by the high amplitude excitation, and converge to the true parameter values. Figure 11 illustrates the

number of resampling events that took place during convergence. Finally, Figure 12 shows histograms of the samples obtained when $t = 2000$.

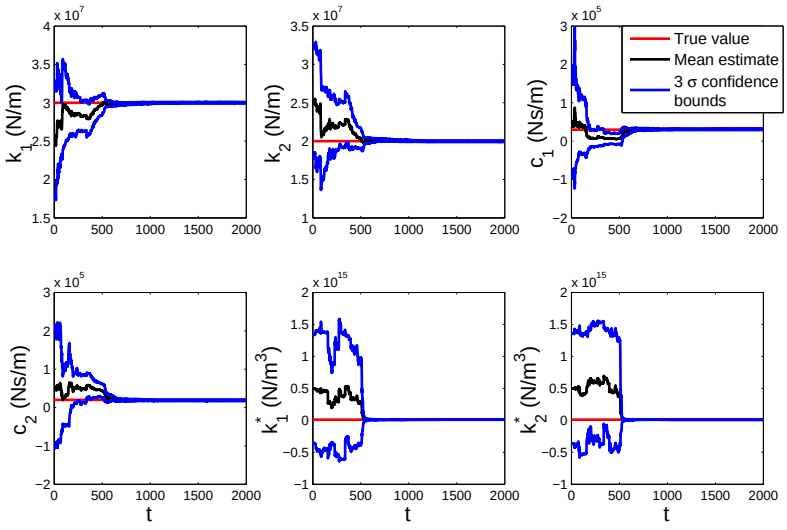


Figure 10: Identifying the parameters of the nonlinear shear building model (equation (45)). Convergence of the parameter estimates towards their true value.

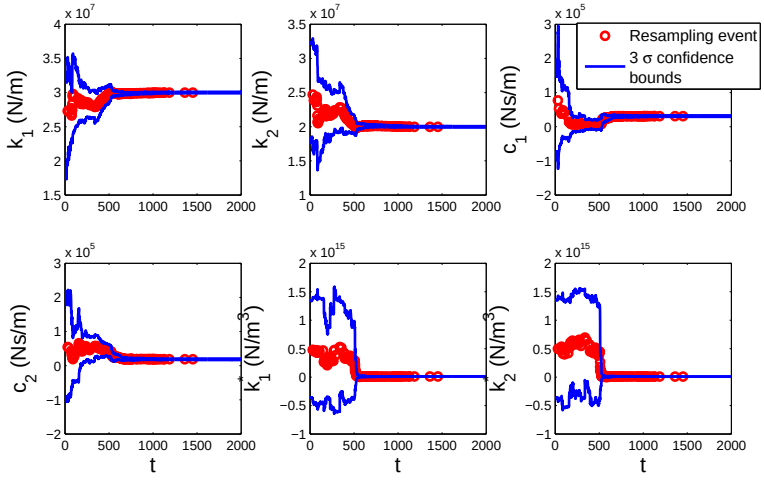


Figure 11: Identifying the parameters of the nonlinear shear building model (equation (45)). Resampling events required during convergence.

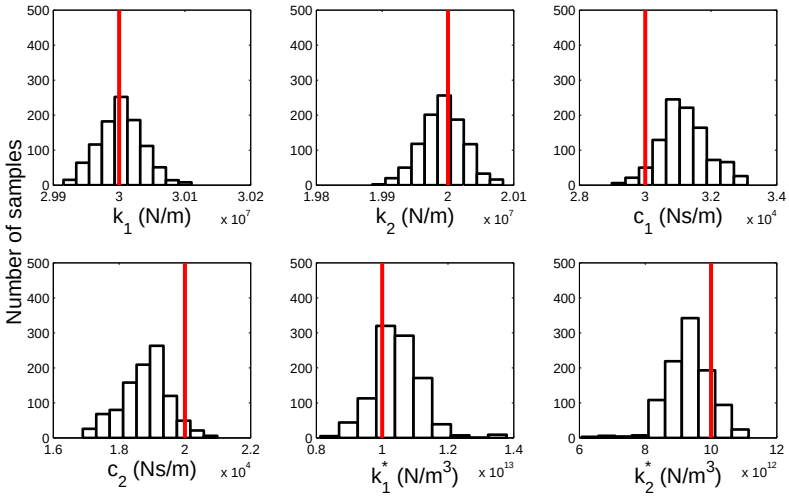


Figure 12: Identifying the parameters of the nonlinear shear building model. Histogram of samples taken at $t = 2000$. Red lines represent true parameter values.

### *5.4. Example - experimental data*

For the sake of completeness, the proposed algorithm was applied to the same experimental data that was investigated in [2], using the Smooth Data Annealing algorithm (an MCMC algorithm which was designed to address the problem tackled in the current paper). A very brief description of the experiment used to obtain this data is given here - further details can be found in [17] while the resulting data set is also available as part of the paper [18] (both of these papers are available Open Access).

The experiment involved the shaker-table test of a rotational energy harvester, which was designed to convert the low frequency translational motion of a 'central mass' into high frequency rotational motion. This translation was achieved using a ball screw, which introduced a significant amount of friction into the system. Following on from the work shown in [17], it was hypothesised that the behaviour of the system could be approximated by the SDOF equation of motion:

$$M\ddot{u} + b\dot{u} + ku + F_c \tanh(\alpha\dot{u}) = -m\ddot{s} \tag{47}$$

where

$$M = m + J\left(\frac{2\pi}{l}\right)^2, \qquad b = \left(\frac{2\pi}{l}\right)c, \tag{48}$$

$u$ is the relative displacement between the shaker table and the central mass, $\ddot{s}$ is the acceleration of the shaker table, $J$ is the system's moment of inertia, $m$ represents the central mass and $l$ is the ball screw lead. It should be

noted that a hyperbolic tangent friction model has been included in equation (47). In the notation of the current paper, the system's inputs, $\boldsymbol{x}_{1:t}$, are time history measurements of the shaker table acceleration while $\boldsymbol{z}_{1:t}$ is the time history measurements of the relative acceleration, $\ddot{u}$. The parameters which require estimation are $\boldsymbol{\theta} = \{c, F_c, \alpha\}$ (it was possible to measure the other parameters of the model directly, to a sufficient level of accuracy).

The first 2000 points of the training data are shown in Figure 13 - it can be seen that testing doesn't begin until $t \approx 400$ and, as a result, one would expect a large influx of information to occur at this point. This is reflected in Figure 14, which shows convergence of the parameter estimates during an initial period (over the first 700 points of training data). The fact that very little resampling was required as the first 400 points were analysed highlights the ability of the proposed method to move efficiently through redundant data. The remaining convergence, until the point where $t = 10000$, is shown in Figure 15. As with the previous results, it is clear that the method proposed here can be used to effectively establish when enough training data has been utilised.

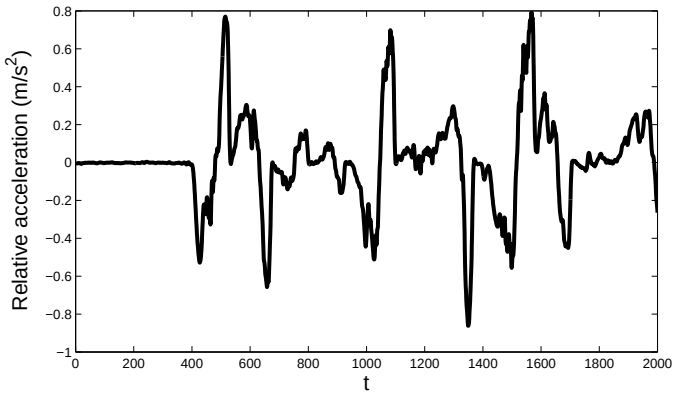


Figure 13: Experimental training data.

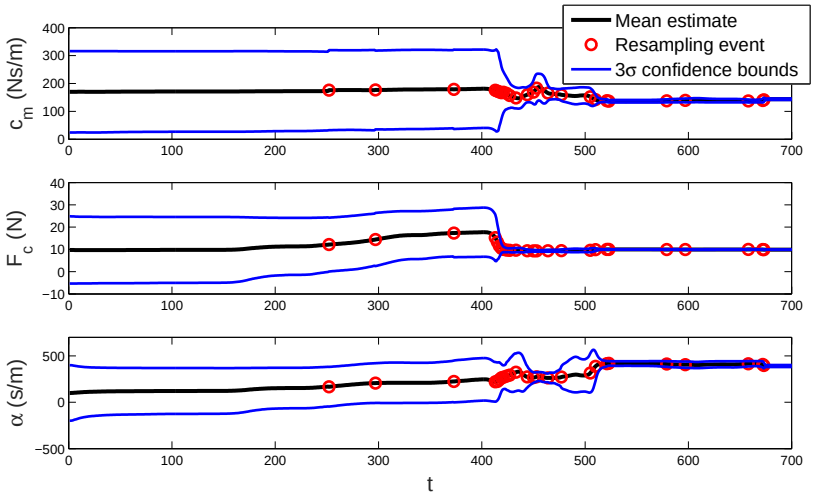


Figure 14: Parameter estimation of the experimental system described in [17] (focused on initial convergence).

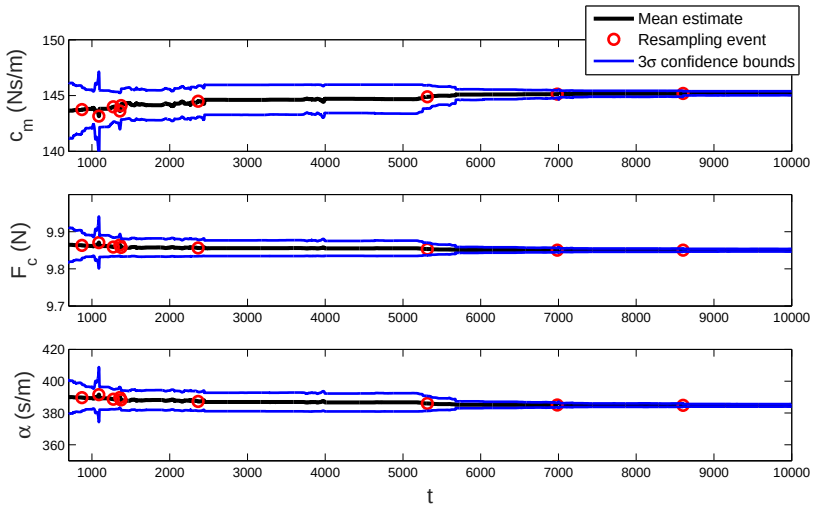


Figure 15: Parameter estimation of the experimental system described in [17] (focused on latter convergence).

Finally, using the parameter estimates realised when $t = 10000$, a Monte Carlo simulation was then used to propagate the resulting parameter uncertainties into an ensemble of model predictions. Figure 16 shows the statistics of the model predictions, alongside measurements of the system's actual response (note that these measurements were not used during the parameter estimation process).

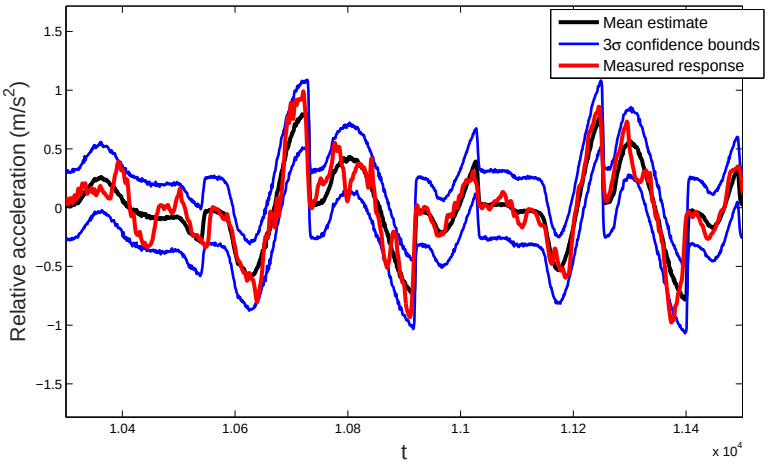


Figure 16: Model predictions after propagation of parameter uncertainties.

## 6. Discussion

### *6.1. Move kernels*

In the current paper, when resampling was required, a SMC sampler was used to facilitate 'movement' in the current set of parameter samples. It is noted here that this can be achieved using various other methods, including MCMC. One could, for example, choose to employ a 'move kernel' similar to that utilised in the TMCMC algorithm [15]. Briefly stated, this would involve generating a set of Markov chains from the set of existing samples, where the probability that each chain will 'grow' is given by the normalised importance weight of the original sample. The important point to note is that the ability of the TMCMC move kernel to exploit parallel processing is dependent on the distribution of importance weights. In an extreme example, where every particle has a weight of zero except for one, the TMCMC move kernel would lead to the development of a single Markov chain, which is impossible to parallelise. The approach described here, however, will *always* be able to fully exploit the benefits of parallel processing (see [19] for details on how this can be implemented on fine-grained parallel architectures). Current studies by the authors [20] show that, in terms of convergence, the two move kernels are able to achieve very similar performance - the power of the move kernel presented here lies in its ability to guarantee parallelisation.

### *6.2. Optimisation*

There are several aspects of the proposed algorithm which require 'tuning', and which influence its computational efficiency. Throughout this paper, for example, the effective sample size was defined as $N/2$. A little

experimentation by the authors showed that this was actually rather conservative, and that by setting an effective sample size of $N/4$, similar results could be achieved at a reduced computational cost. The size of the proposal distribution, in addition, strongly influences the number of times that resampling needs to occur. The optimal selection of these 'settings' is an aspect which the authors intend to look into as part of future work - the purpose of the current paper is to establish a fundamental methodology, whose applicability to computationally expensive models will increase alongside continued advancements in parallel computing technology.

### *6.3. Future work*

The method proposed here relies on the assumption that the likelihood can be written as a product of independent distributions (equation (10)). Such an assumption prevents the method from being applied directly to, for example, estimating the hyperparameters of Gaussian processes. For future work the authors aim to address this restriction. Gaussian processes are of particular interest because of their apparent suitability with regard to the quantification of model error [21] (something which has not been specifically addressed in the current paper).

## 7. Conclusions

In this paper the authors present a method which can be used to efficiently 'track' how one's estimates of a system's parameters vary (and converge) as an increasingly large set of training data is analysed. The method, which utilises a Sequential Monte Carlo sampler, is able to efficiently exploit the inevitable redundancy in large sets of measurements and, by being suitable for parallel processing, is suitable for modern computing architectures. The ability of the method to aid the estimation of dynamical systems from data which is noisy, data which arrives in 'blocks' or data where large influxes of information can occur, was demonstrated using simulated and experimental case studies.


[1] P.L. Green. Bayesian system identification of a nonlinear dynamical system using a novel variant of simulated annealing. *Mechanical Systems and Signal Processing*, 52:133–146, 2015.

[2] P.L. Green. Bayesian system identification of dynamical systems using large sets of training data: A MCMC solution. *Probabilistic Engineering Mechanics*, 42:54–63, 2015.

[3] P.L. Green, E.J. Cross, and K. Worden. Bayesian system identification of dynamical systems using highly informative training data. *Mechanical systems and signal processing*, 56:109–122, 2015.

[4] W. Betz, I. Papaioannou, and D. Straub. Transitional Markov Chain Monte Carlo: Observations and Improvements. *Journal of Engineering Mechanics*, 142(5):04016016, 2016.

[5] L. Kulakova, P. Angelikopoulos, P.E. Hadjidoukas, C. Papadimitriou, and P. Koumoutsakos. Approximate Bayesian Computation for Granular and Molecular Dynamics Simulations. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, page 4. ACM, 2016.

[6] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.

[7] A. Kong. A note on importance sampling using standardized weights. *University of Chicago, Dept. of Statistics, Tech. Rep*, 348, 1992.

[8] A. Kong, J.S. Liu, and W.H. Wong. Sequential imputations and Bayesian missing data problems. *Journal of the American statistical association*, 89(425):278–288, 1994.

[9] J. Ching, J.L. Beck, and K.A. Porter. Bayesian state and parameter estimation of uncertain dynamical systems. *Probabilistic engineering mechanics*, 21(1):81–96, 2006.

[10] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *Signal Processing, IEEE Transactions on*, 50(3):736–746, 2002.

[11] N. Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

[12] P. Del Moral, A. Doucet, and A. Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[13] S. Maskell. An application of sequential monte carlo samplers: an alternative to particle filters for non-linear non-gaussian sequential inference with zero process noise. *Proccedings of the IET Data Fusion and Target Tracking Conference*, 2012.

[14] T. Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2):185–194, 1995.

[15] J. Ching and Y.C. Chen. Transitional Markov chain Monte Carlo method for Bayesian model updating, model class selection, and model averaging. *Journal of engineering mechanics*, 133(7):816–832, 2007.

[16] S.K. Au, F.L. Zhang, and P. To. Field observations on modal properties of two tall buildings under strong wind. *Journal of Wind Engineering and Industrial Aerodynamics*, 101:12–23, 2012.

[17] P.L. Green, M. Hendijanizadeh, L. Simeone, and S.J. Elliott. Probabilistic modelling of a rotational energy harvester. *Journal of Intelligent Material Systems and Structures*, page 1045389X15573343, 2015.

[18] P.L. Green and K. Worden. Bayesian and Markov chain Monte Carlo methods for identifying nonlinear systems in the presence of uncertainty. *Phil. Trans. R. Soc. A*, 373(2051):20140405, 2015.

[19] S. Maskell, B. Alun-Jones, and M. Macleod. A single instruction multiple data particle filter. *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, pages 51–54, 2006.

[20] P.L. Green and S. Maskell. Parameter estimation from Big Data using a Sequential Monte Carlo sampler. *Proceedings of USD 2016, Uncertainty in Structural Dynamics*, 2016.

[21] M.C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.