McMaster University

PhD Comprehensive Examination Report

Parameter Estimation for Nonlinear Stochastic Dynamical Systems

Author : David Champredon
School of Computational Science and Engineering

Examining Committee : Ben Bolker
Jonathan Dushoff (sup.)
David Earn (sup.)
Marek Smieja

April 27th, 2015

McMaster University
1280 Main Street West
Hamilton, Ontario – Canada

Contents

1 Introduction

Mathematical models describing a real-world system are ubiquitous. Their purpose is to disentangle and quantify the often complex dynamics at play.

Let's label $\mathcal{X}$ the real-world quantities we are interested in. Often, $\mathcal{X}$ is affected by independent random environmental events such that its value cannot be reasonably described with a purely deterministic model. Stochastic variables are introduced in the model such that this uncertainty, called *process error*, can be taken into account. Given a set of model parameters θ, the mathematical model produces an output value for $\mathcal{X}$. If $\mathcal{X}$ cannot be measured with accuracy or is simply not observable, introducing a measurement or observation model enables to "link" the model to actual observations. Now the model explicitly takes into account the *observation error* by providing (given a value for $\mathcal{X}$) an output, labelled $\mathcal{Y}$, that can be directly compared with observed data. Hence we have the following flow:

$$\theta \rightarrow \text{Model} \rightarrow \mathcal{X} \rightarrow \mathcal{Y} \rightarrow \text{data}$$

Often, the model involves parameters we don't know the value *a priori*. So, we are interested in reversing this flow: given the data observed, what are the parameter values θ^* that makes the model best explains these observations. Because the model is stochastic, one possible way is to find the value of θ that maximizes $L(\theta, D)$, the likelihood of observing the data D. Mathematically, it is formulated as

$$\theta^* = \text{argmax}_\theta \, L(\theta, D) \tag{1}$$

Another possibility is to consider the model parameter as random variables Θ and aim at calculating the expected value of $g(\Theta)$, for some pre-specified function g and again given the data D:

$$E\left(g(\Theta)|D\right) = \int g(\theta)\; p(\theta|D) d\theta \tag{2}$$

When the model is not "too complex" and the number of parameters to fit (dimension of θ) is not too large, solving equations (1) or (2) may be achievable without too much difficulties. Unfortunately, practical issues quickly arise because models describing the system of interest are typically complex and usually have several parameters to fit. This results in a likelihood surface (1) where the global maximum is difficult to locate and/or a high dimensional integral (2) that cannot be numerically evaluated, even with powerful computers.

This reports reviews some of the latest state-of-the-art statistical methods to estimate parameters of nonlinear stochastic dynamical systems.

2 Review

2.1 Particle filter

Importance sampling

Before describing the particle filter, let's briefly define its fundamental building block, importance sampling. Suppose X is a random variable with density ϕ and we would like to evaluate

$$E(g(X)) = \int g(x)\phi(x)dx$$

but we don't know how to sample from ϕ. However, we know a distribution s we can easily sample from. Assume that $s > 0$ wherever $\phi > 0$. We can then write

$$\begin{aligned} E(g(X)) &= \int g(x)\phi(x)dx \\ &= \int g(x)\frac{\phi(x)}{s(x)}\, s(x)dx \\ &= \int g(x)w(x)\, s(x)dx \\ &= E_s(g(X)w(X)) \end{aligned}$$

So this expectation is calculable by sampling from the probability density s and performing a standard Monte Carlo, but adjusting the expectation by the correcting factor $w(x) = \phi(x)/s(x)$ because we do not use the "natural" density ϕ. We can write

$$\phi(x) \propto w(x) \times s(x) \tag{3}$$

Particle Filtering

Particle filtering (also known as sequential Monte Carlo, bootstrap filter,survival of the fittest) is designed for iterative estimations of partially observed (or "hidden") Markov processes (POMP). Conceptually, defining a POMP is straightforward: there is a "true", unobserved stochastic Markovian process X_t that can only be inaccurately measured through a variable Y_t. Hence, only three components are needed to fully specify a POMP: *i*) a starting condition X_0, *ii*) a description of how the true process evolves to the next time step:

$$X_{t+1} \sim f_1(X_t, \alpha)$$

iii) a description of the observation process at each time step:

$$Y_t \sim f_2(X_t, \alpha)$$

where f_i are some known distributions and α a vector of fixed parameters. Given observations $Y_{1:t} = Y_1, ..., Y_t$ we would like to sample from the target posterior distribution $p(X_t|Y_{1:t})$. Using both the Markov property and Bayes rule we have,

$$p(X_t|Y_{1:t}) \propto p(Y_t|X_t) \times p(X_t|Y_{1:t-1})$$

Because we can sample from $p(X_t|Y_{1:t-1})$ which is by definition f_1, using equation (3) above suggests an importance sampling method using $p(X_t|Y_{1:t-1})$ as the sampling distribution and the likelihood $p(Y_t|X_t)$ as the correcting factor.

Assume an initial distribution for X_0 that gives a starting samples ("particles") set. They are resampled *with replacement* according to weights proportional to the likelihood of observing these samples. These resampled particles $\tilde{X}_0$ are then used to predict the next time step $X_1 = f_1(\tilde{X}_0)$. Then, X_1 is used as the starting point for the next iteration, and so on. We have just described the "particle filtering" algorithm[4], which is formally described in Algorithm 1 and illustrated in Figure 1. This method constructs, at every time step t, a set of N particles that approximate with a *discrete* distribution the *continuous* posterior distribution at time t, that is $p(X_t|Y_{1:t})$:

$$p(X_t|Y_{1:t}) \simeq \frac{1}{N} \sum_{i=1}^{N} \delta_{X_t^{(i)}}(X_t) \tag{4}$$

where $(X_t^{(i)})_{i=1..N}$ are the N filtered particles at time step t and δ the Dirac measure. Furthermore, that estimation is *asymptotically unbiased* (as $N \to \infty$).

The step involving resampling with replacement is actually optional and could be omitted. However, such an algorithm is likely to be less efficient because the particles would spread more thinly resulting in a a larger variance for the estimated distribution. But there is a drawback when resampling: dependence among the particles is introduced. The resampling itself is actually generated by the discrete probability (4): the value of a sample associated to a large weight will be drawn many times. Hence, the resampling step reduces the diversity of sampled values: this is know as "sample impoverishment". Solutions to this problem have been proposed to introduce (randomly or not) small perturbations after the resampling step, or using a continuous approximation to the resampling discrete density ("regularized" particle filter)[19, 22].

2.2 Iterated filter

Iteration and data cloning

The iterative nature of the methods presented below is similar to the "data cloning" presented by Lele *et al.*[18]. The idea of data cloning is very simple: pretend that the observation data set is replicated (cloned) M times. By doing so, the likelihood $L(\theta)$ is exponentiated to M, that is we deal with $(L(\theta))^M$. If M is large enough, the global maximum of L will clearly stand out from any other local extremum, facilitating its identification. This is illustrated in Figure 2. Also, when used in a Bayesian analysis, data-cloning is insensitive to the prior because the predominant peak dwarfs any prior density and concentrates the posterior at the global maximum likelihood [18].

Iterative Filter 1

This method (labeled IF1 here) was proposed by Ionides et al [14] for POMP models. The underpinning idea is to augment the model changing the *constant* parameters θ into a time-dependent stochastic process $\theta(t)$, typically a gaussian random walk. The authors proved that the iterative method proposed converges to the maximum likelihood estimator as the variance of the stochastic process defining $\theta(t)$ tends to zero.

Let's first give a high level description of IF1. Instead of performing a single particle filter from the observed data $y_{1:T}$ to estimate the posterior distribution of unobserved variables (X_t) at each time step, IF1 iterates several times this process. At each filtering iteration m, a summary statistic $\theta^{(m)}$ is calculated based on the mean and variance of the"local" estimations of parameter θ performed at each time step. By construction, the variance involved in calculating $\theta^{(m)}$ decreases at each iteration. This summary statistic is then used from one iteration to the next and becomes less and less stochastic (because of a pre-specified decreasing variance) and converges to the maximum likelihood estimator.

Now, let's take a detailed look at IF1 which is outlined in Algorithm 2 and an illustrative example is given in Figure 3. The process function is labelled f and the observation function g.

As mentioned in the high level description, IF1 iterates several times particle filtering on the whole time series of observations $y_{1:T}$. Lines 6 to 18 in Algorithm 2 merely implement a single standard particle filter. The top panel of Figure 3 simply depicts the observed data time series. The second panel describes the first iteration of IF1.

Iteration 1. Given a pre-specified value $\theta^{(1)}$ for θ, J particles (samples) are drawn from a normal distribution centred at $\theta^{(1)}$ and variance $b\Phi$, where Φ is the pre-specified covariance matrix of parameters components θ and b adjusts the initial sampling variance. In the second panel of Figure 3, $\theta^{(1)}$ is illustrated by the large red diamond on the y-axis and the $J = 5$ particles initially drawn by yellow open circles. Based on the first observation y_1 and our knowledge on both the true process dynamic f and observation process g, the particle filter is performed at time $t = 1$.

At each time step t, the mean $\bar{\theta}_t$ and variance V_t of the J particles is recorded (represented by solid black squares for the mean and vertical segments for the variance in Figure 3 ; line 19, Algorithm 2). Values of $\bar{\theta}_t$ and V_t at every time step are then used to update the (current iteration) mean estimate $\theta^{(1)}$ to the next one $\theta^{(2)}$ using the recursive formula (for each element of vector θ, algorithm line 24)

$$\theta^{(m+1)} \leftarrow \theta^{(m+1)} \sum_{t=1}^{T} \frac{V_1}{V_t}(\bar{\theta}_t - \bar{\theta}_{t-1}) \quad (5)$$

In other words, Equation (5) updates the value of the next estimate $\theta^{(m+1)}$ by adding the contribution of local (at every time steps) ones weighted by their precision (inverse variance). This is the core formula underpinning IF1. This update is depicted at the rightmost of the second panel of Figure 3, where the updated value $\theta^{(2)}$ is another red solid diamond.

Iteration 2. The second iteration is very similar to the previous one. The main difference is that the starting mean parameter value at time $t = 0$ is now centred at $\theta^{(2)}$ and the particles are sampled with a smaller variance $a\Phi$, where $a < 1$ is the so-called cooling parameter that decreases sampling variance at each main iteration (*not* in time). This is highlighted in the third panel of Figure 3 where the black vertical segments V_t are shorter than the ones in the previous iteration (second panel).

Last iteration M. After the particle filters (across the full time series of observations) have been iterated M times, the particles are now very close to one another thanks to the decreasing variance (Figure 3, bottom panel). Hence, within that last iteration, local estimates $\bar{\theta}_t$ are barely changing and the variance V_t is very small (very short vertical segments). So the very last updated value $\theta^{(M+1)}$ is not expected to change much and is taken to be the final estimation maximizing the likelihood.

Iterative Filter 2

The same authors of IF1 recently proposed a new method, labelled IF2, supposed to be an improvement [15]. Although both IF1 and IF2 are iterative particle filters that aim to find parameters associated with the maximum likelihood, they have noticeable differences. The summary statistics ($\bar{\theta}$ and V) computed in IF1 at every time step in order to update the parameter estimate for the next main iteration are dropped in IF2. Instead, IF2 introduces a stochastic perturbation h at every time step (typically, h could be a gaussian multivariate). Unlike IF1, the result of IF2 will be J particles (samples) of parameters θ that should be close to the likelihood maximum. IF2 method is outlined in Algorithm 3 and illustrated in Figure 4.

Iteration 1: The initial guess parameter $\Theta^{(0)}$ is first perturbed before being used for the very first prediction of $X_F(0, .)$ to get the starting parameter $\theta_F^{(1)}(0, .)$. The yellow solid circles represent the J particles composing $\theta_F^{(1)}(0, .)$ in Figure 4. Each particle of $\theta_F^{(1)}(0, .)$ is then perturbed at time 0 by resampling it from the density h which has mean $\theta_F^{(1)}(0, .)$ (line 8, Algorithm 3). The perturbed particles $\theta_P^{(1)}(0, .)$ are represented with purple solid circles. Then, these perturbed particles go through the particle filter, generating J new particles $\theta_F^{(1)}(1, j)$ for time $t = 1$. Note that *both* parameter θ and state process X are *jointly* resampled in the particle filter (algorithm line 14). At the time horizon T, we end up with J particles $\theta_F^{(1)}(T, .)$ (solid red circles) that will be re-used as starting condition for the next step.

Iteration 2: This is the same process as described in iteration 1. The only difference is that the perturbations can be smaller than the ones at the previous iteration if the sequence σ_m is chosen to decrease (which is likely the case in practice). This is why particles are illustrated with less variance in iteration 2 in Figure 4 (third panel). The last set of J particles at time T will be again re-used for the third iteration, and so on.

Last iteration M: If enough iterations are run (M large enough), the final set of J particles $\Theta^{(M)} = \theta_F^{(M)}(T, .)$ should converge to the maximum likelihood estimator.

Note that when $M = 1$ and no perturbation, IF2 is simply a standard particle filter.

2.3 Markov Chain Monte Carlo

Both iterated filtering methods IF1 and IF2 aim to find the maximum likelihood estimate defined in (1). Now, let's consider a method to approximate the expectation (2) of the parameters θ given observed data D.

Let's rewrite (2) by first applying Bayes' formula and then the countable probability property:

$$E(g(\Theta)|D) = \frac{\int g(\theta)\, p(D|\theta)p(\theta)d\theta}{\int p(D|\theta)p(\theta)d\theta} \quad (6)$$

The advantage of this Bayesian formulation is now we consider densities that can be easier to evaluate: likelihood $p(D|\theta)$ and some prior knowledge (if any) about the distribution of θ. Hence our distribution of interest is now $\phi(\theta, D) = p(D|\theta)p(\theta)$. In some specific low-dimensional cases, analytical or a simple numerical integration of (2) may be possible. But most applications typically involve dimensions and posterior densities that quickly make numerical integration not feasible. High dimensionality can be tackle with a basic Monte-Carlo (MC) technique. Indeed, probability theory guarantees convergence of the MC estimator to our integral:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} g(\theta_i) = E(g(\Theta)|D) \quad (7)$$

where θ_i are samples drawn independently from the posterior ϕ. MC saves us against high-dimensionality because the number N of samples is, in theory, independent from the dimension of Θ. But an ordinary MC integration may not be feasible, because drawing *independent* samples from the distribution $\phi(\theta)$ can be too costly or impossible.

When sampling from ϕ is not possible, sampling using a Markov chain can help. Indeed, if X_t is a Markov chain converging to the invariant distribution ϕ, it can be shown that

$$\bar{g} = \frac{1}{n-m+1} \sum_{t=m}^{n} g(X_t)$$

is a good estimator of $E(g(\Theta))$ for appropriate values of m (burn-in period) and n (total number of iterations) with $m < n$.

Assuming we know explicitly the likelihood and the prior, then we know ϕ and need to construct the Markov chain X that will converge to ϕ. Note that, contrary to the canonical Monte Carlo method, samples from the Markov chain will *not* be independent. It's not a problem because what we ultimately care about is that their (stationary) distribution is ϕ.

Metropolis-Hasting

The Metropolis-Hasting (MH) algorithm provides a convenient way to construct a Markov chain X converging to a desired distribution. If X_t is the current state of the chain X at time t, we want to know its next value at time $t+1$. This algorithm, proposed by Metropolis and Hasting [**?**] ensures X will eventually converge to ϕ:

- Draw a candidate value Y from a distribution labelled $q(.|X_t)$
- Accept Y with probability

$$p_{acc} = \frac{\phi(Y)q(X_t|Y)}{\phi(X_t)q(Y|X_t)} \wedge 1 \quad (8)$$

A formal algorithm is given in appendix (Algorithm 4). The strength of this algorithm is that it does not depend on the initial condition X_0 (ergodic theorem for Markov chains) and the proposal distribution $q(.|.)$ can be *any* distribution. So MH is fairly general because the proposal distribution q is left to be chosen. The actual relationship between the target distribution ϕ and the proposal q influences the efficiency of a MCMC algorithm. Hence a lot of efforts have been dedicated

in choosing smart proposal distributions, resulting in a swarm of MCMC samplers (covering them all is beyond the scope of this report). Briefly, here are some well-known choices of proposal distributions within a MH context:

Metropolis: When the proposal is symmetrical, $q(x|y) = q(y|x)$, the acceptance probability simplifies to the ratio of target densities. An example is a proposal that only depends on the distance between the current state value and the new proposal (random walk).

Gibbs: Each element of X can be updated independently with the full conditional distribution, that is

$$q(Y_t^{(i)}|X_t) = q_i(Y_t^{(i)}|X_t^{-(i)})$$

where $Y_t^{(i)}$ is the i^{th} element of Y_t and $X_t^{-(i)}$ is X_t stripped out of its i^{th} element. A remarkable consequence of this proposal distribution is that all candidates are *always* accepted ($p_{acc} = 1$).

Independence: The proposal for the next move can be independent from the current position: $q(Y|X) = q(Y)$.

Slice sampling: This sampler is an alternative to MH-like algorithm and is based on a standard rejection sampling coupled with a pre-specified width of exploration ("slice")[**?**].

Hamiltonian MCMC

One drawback of Metropolis-Hasting sampling is its random-walk nature: proposals for the next sample do not take into account the topology of the target distribution. As a consequence, parameter space may be poorly and slowly explored.

A major improvement is achieved with so-called Hamiltonian MCMC[20]. The main idea is to take into account the topology of the target distribution when proposing a new value for the next time step of the Markov chain. Using naively the gradient of the target distribution is unlikely to lead to a successful algorithm because conditions for the Markov chain to converge to the target distribution can be unsatisfied. But, when using techniques borrowed from Hamiltonian dynamics this methods turns out to perform well.

First, Hamiltonian MCMC take the counter-intuitive approach to double the dimension of the parameter space "positions" $\theta = (\theta_1, ..., \theta_d)$ by introducing d auxiliary variables $r = (r_1, ..., r_d)$ representing the "moments" of each dimension. This is done to mimic the physics framework. Our target distribution ϕ is rewritten as

$$\phi(\theta) = e^{-U(\theta)}$$

with $U(\theta) = -\log(\phi(\theta))$ representing the "potential" energy of our system. Then, a "mass" matrix M is arbitrarily chosen to define the kinetic energy of our system:

$$K(r) = \frac{1}{2} r^T M^{-1} r$$

It is possible – and actually often the case – to simply choose M as the identity matrix. Using a non-identity mass matrix is equivalent to transforming linearly the parameter space [21] and can be a way to take into account correlation between parameters. Finally, a distribution sampling *both* θ and r is defined:

$$A(r, \theta) \propto \exp(-H(\theta, r))$$

where $H(\theta, r) = U(\theta) + K(r)$ is the so-called Hamiltonian of the system. Note that r and θ are independent under A. Again, r samples are just auxiliary and will be ignored. We are ultimately interested in samples for θ. That seems artificial, especially because of the independence, but the link between r and θ is made through the Hamiltonian dynamic.

To propose samples from A, the Hamiltonian dynamic is simulated:

$$\begin{aligned} d\theta &= M^{-1} r \, dt \\ dr &= -\nabla U(\theta) \, dt \end{aligned} \tag{9}$$

Some technical notes about system (9): it conserves the Hamiltonian (H constant) so the sample proposal will always be accepted (in theory); it is reversible in time (one-to-one mapping) so the distribution A is left invariant ; it preserves volume, hence no correction involving Jacobian needed for calculating acceptance probability.

If the Hamiltonian dynamics (9) is naively implemented with a standard Euler scheme, it is likely that instabilities arise. A "leapfrog" (Stormer-Verlet) algorithm is a better choice because more robust. But this robustness comes at the price of not conserving the Hamiltonian which means a Metropolis-Hasting correction is needed, using an acceptance probability $p_{acc} = \exp[H(\theta', r') - H(\theta, r)]$ where θ' is the new proposal. The leapfrog algorithm in its basic version needs both the number and size of parameter space steps (ϵ and L in Algorithm 5) to be specified when integrating the Hamiltonian dynamics (9). This can be a problem in practice.

Assuming the step size and trajectory length are appropriately tuned, Hamiltonian MCMC is likely to explore more efficiently the parameter space of the targeted distribution than a standard Metropolis-Hasting MCMC. The cost of a random walk Metropolis per independent sample from a distribution in dimension d is $O(d^2)$ whereas Hamiltonian Monte Carlo stands at $O(d^{4/5})$ [13].

As a final note, because the gradient of the target distribution is involved, Hamiltonian MCMC is reserved for *continuous* distributions.

NUTS in HMCMC

Despite the improved stability provided by the leapfrog method, a lot of "fine tuning" with the space step size ϵ and the length of the trajectory ϵL the dynamics are integrated upon. This is where most of the difficulty is in practical applications. Ideally, we want a Markov chain that explores efficiently the parameter space where the mass of the target distribution is: this exploration must be fast and relatively exhaustive. Indeed, a balance must be reached:

- if ϵ is too small, we have too fine a resolution which wastes computational powers (slow)
- if ϵ is too large, the resolution is too coarse and each proposed move is more likely to be rejected (again slowing things down by wasting computational power)
- if L is too small, the progress of exploration is too slow because successive samples are too close from one another
- if L is too large, Hamiltonian trajectories can "loop back", that is end up close to the previous sample and hence resulting in a slow (inefficient) progression.

The main idea of the No U-Turn Sampler (NUTS, [13]) is to monitor the trajectory of the Hamiltonian dynamics and stop it when it starts to traceback its steps (hence avoiding a U-turn). That essentially controls for the number of steps L, for a given step size ϵ.

There are many conditions an algorithm must satisfy to keep all the desired properties of the Markov chain, most importantly its ergodicity (its ability to converge to the unique target distribution) and avoiding a random-walk behaviour. Hence, naively imposing a stopping condition on the Hamiltonian dynamics trajectory is likely to break those nec-

essary conditions. This is why NUTS takes great care of making sure all desirable conditions are satisfied and elaborate a relatively complex algorithm for that purpose. It is a relatively complex and technical algorithm and only an outline is described here. See Figure 5 for an illustrative example.

• **Backward/Forward integration:** In order to guarantee detailed balance (that is time-reversibility, hence convergence to the correct target distribution), the Hamiltonian dynamics (9) is integrated by blocks of 2^j leapfrog steps, backward and forward in time. The time direction choice of the jth block is randomly made (Bernoulli). Irrespective of the direction, the 1st block will have $2^0 = 1$ step integrated, the 2nd block $2^1 = 2$ steps and so on, the jth block will have 2^{j-1} steps.

• **Stop Criteria:** This trajectory construction by blocks of 2^j steps is stopped whenever one of these two condition is met:

No U-Turn: If the distance between the starting position of the last block (say θ_0) and the latest integrated position (say θ_n) reduces. This is tested by checking if the scalar product $(\theta_n - \theta_0).r_n$ is negative.

Low probability: If the area explored has a very low probability with respect to the target distribution. This means the algorithm would otherwise have to deal with extremely large errors (recall we deal with the log-density) leading to erratic behaviour.

• **Candidates set:** Among all positions generated by the trajectory blocks construction, candidate positions are identified such that detailed balance is satisfied. In particular, if one position violates detailed balance, the whole block is ignored. This might seem inefficient (calculating 2^j leapfrog positions for nothing) but that is the price to pay for guaranteed ergodicity.

• **Random candidate selection:** Once all candidate positions have been identified, one is pick at random (uniformly) to be the next sample proposal.

The NUTS algorithm deals with the *length* of the trajectory leading to a proposed sample. That is, it provides a way to automatically set ϵL. But that still leaves both parameters ϵ and L unspecified (only their product is). Hoffman and Gelman [13] actually provided a method to appropriately set step sizes for both the Hamiltonian MCMC and NUTS algorithms (both labeled ϵ) during the warm-up phase of the HMC. Jumping over technical points, the main idea is to take advantage of the warm-up phase of the HMC (phase where samples will be ignored) to fit the step sizes such that we experience a pre-defined acceptance rate during this phase.

2.4 Exact-approximate MCMC

We saw that standard Metropolis-Hasting MCMC offers a flexible way to construct a Markov chain converging towards a desired target (posterior) distribution ϕ. This algorithm relies on the acceptance probability (8) and more specifically on being able to evaluate explicitly ϕ. So it seems Metropolis-Hasting MCMC cannot be applied if we are dealing with a model that does not allow us to evaluate explicitly ϕ.

But what if we could approximate the distribution with some kind of numerical method $\hat{\phi} \approx \phi$ and plug that approximation back in the probability acceptance formula (8)? This acceptance probability would become

$$p_{acc} = \frac{\hat{\phi}(Y)q(X_t|Y)}{\hat{\phi}(X_t)q(Y|X_t)} \wedge 1 \tag{10}$$

This is tricky business, because approximating our target distribution ϕ may break the ergodicity of the Markov chain we are constructing. In other words, it is not obvious that with this modification the Markov chain

will still converge to our target distribution. Andrieu *et al.*[3] have shown that it is indeed possible to use such approximation without breaking the ergodicity, provided that – among other weak conditions – the distribution approximation is *unbiased*:

$$E(\hat{\phi}) = \phi$$

Hence, even using an *approximate* target distribution we are still able to construct a Markov chain that converges to the *exact* distribution. Such possibility to carry out MCMC without being able to evaluate explicitly the target distribution extends the range of applicability of MCMC methods.

An illustrative example is given in Appendix Listing 1 where the distribution was not formally estimated but simply perturbed with some random noise to mimic error from a genuine estimation. Its output is shown in Figure 6 where the MCMC using the perturbed target also converges to the desired distribution.

A particular case is particle MCMC introduced by Andrieu *et al.*[2]. Because particle filtering provides an unbiased estimate of a given target distribution, it is an natural candidate for $\hat{\phi}$.

2.5 MCMC diagnostics

When performing inference with MCMC, the Markov chain was constructed satisfying *necessary* conditions for its convergence, but unfortunately there is no *sufficient* conditions available. Hence, we have to rely on several diagnostics to assess the actual convergence of the chain. Because the diagnosis step is one of the most important in practice, the most popular diagnostic methods are briefly reviewed here. The Markov chain is noted X_t and we assume a pre-specified amount of iterations has already been discarded (burn-in or warm-up period). No single diagnostic test can definitely assess convergence but several of them may indicate convergence has likely been reached, or not.

The eye Before running any statistical test, a simple visual inspection of a plot of the chain values can assess obvious convergence problems.

Geweke[9] The earliest values of the chain is compared with the latest part (typically the first 10% vs. the last 50%) using a two-sided T-test. If the chain has reached a stationary distribution, then both ends should have the same mean.

Gelman Rubin ($\hat{R}$)[8] Several chains with different starting value are considered. If all chains have converged to the target distribution, the variance *between* the chains should be small compared to the variance *within* the each chain. A statistic, $\hat{R}$, representing this statement is designed such that convergence is likely reached when $\hat{R} = 1$.

Autocorrelation If convergence has been reached, then the auto-correlation (for a pre-specified lag) of a chain should be consistently close to 0.

Effective sample size Another measure based on auto-correlation is the effective sample size which calculates the equivalent number of samples bearing independent information. Mathematically,

$$ESS = N/(1 + 2\sum_{1}^{N} \rho_k)$$

with N the number of iterations run and ρ_k the autocorrelation with lag k. An ESS value close to N is a good sign of convergence.

3 Applications

3.1 Generation interval distribution

Generation intervals – the difference between the time of infection of an infectee and his/her infector – are fundamental quantities when modelling epidemics. Unfortunately they are hardly observable. Serial intervals – the difference between the time of symptoms onset of an infectee and his/her infector – are often used as proxy and often estimated through contact-tracing. But contact tracing can sometimes be unreliable (recall bias, medical-staff shortage, etc.) and correlation between serial and generation intervals may not be obvious for some diseases. Hence it would be useful to estimate generation interval via raw incidence data.

First, let's take the hypothetical example of a disease that has a generation interval of a fixed length, say 5 days, for all individuals. So, by simply looking at the incidence time series we would be able to determine the generation interval because the successive waves of infections would clearly stand out 5 days apart. Now, if there actually were small variations around the fixed-length generation interval, we would probably still be able to identify its mean length. This simplified example leads to the idea of trying to estimate a generation intervals directly from incidence time series.

Discrete Model Let I_t be the observed incidence at time t and U_t the real (unobserved) incidence. The observation of incidence is not deterministic, but driven by the following relationship

$$I_t \sim \text{Binom}(U_t, \rho) \tag{11}$$

where ρ is the average reporting rate that I assume constant here.

The unobserved real incidence U is supposed to be driven by the following renewal equation:

$$U_t = \sum_{k=1}^{T} \lambda(k)\, U_{t-k} \tag{12}$$

where $\lambda(k)$ is the force of infection of the cohort infected k time units ago and T is the largest possible value for the generation interval. So $\lambda(k) = 0$ for all $k > T$ and I assume $T > 1$. I assume we observe the beginning of an outbreak in a large population so susceptible depletion can be ignored. The basic reproductive number at time t is $\mathcal{R}_0 = \sum_k \lambda(k)$ and the generation interval distribution will be given by $g(k) = \lambda(k)/\mathcal{R}_0$. But the epidemiological process is not deterministic, so the real incidence is assumed Poisson distributed:

$$U_t \sim \text{Poisson}\left(\sum_{k=1}^{T} \lambda(k)\, U_{t-k}\right) \tag{13}$$

Because U depends on its values own values at several past dates, this formulation is *not* Markovian.

Distribution shape If the shape of the distribution λ is *a priori* not known, then each value of λ at different time points can be considered as an independent random variable. A possible modelling is:

$$\lambda(k) \sim \text{Unif}(0, a) \quad k = 1, ..., T \tag{14}$$

with a a fixed constant. Hence, there are T independent parameters to estimate. This modelling framework will be called *non-parameteric* hereafter.

If there is enough information to impose a specific shape to λ, then it can be parameterized with one or more "shape parameters". Here, the incentive is to use as few parameters as possible in order to simplify the inference problem. I arbitrarily chose

$$\lambda(t) = bt^3 e^{-at} \tag{15}$$

where a and b are shape and scale parameters to estimate. Note that b is proportional to $\mathcal{R}_0$.

Inference methods The formulation (13) is non-Markovian. This precludes the use of techniques based on POMP, like iterative filters. However, MCMC methods can be applied in this case. Using Hamiltonian-MCMC will not be possible with the discrete formulation because this methods can only handle *continuous* variables. Hence the discrete formulation in Equation (13) is not appropriate for this method. However, it is conceivable to have a continuous formulation if the population considered is large. In that case, incidences U and I are considered as fractions of the total population (e.g., $U \to U/N$ with N the total population size). The new continuous formulation becomes

$$U_t \sim \mathrm{LN}\left(\sum_{k=1}^{T} \lambda(k)\, U_{t-k}\,;\, \sigma_U\right) \qquad (16)$$

and the observation process

$$I_t = Q_t\, U_t \quad \text{with } Q_t \sim \mathrm{LN}(\rho\,;\, \sigma_Q) \quad (17)$$

The shape assumptions on λ are the same as in the discrete case.

Simulated data In order to test method efficiency, 60 incidence data points were generated using model (13) for the discrete and (16) for the continuous inference using the following benchmark values: $\mathcal{R}_0 = 1.8$, $T = 10$, $a = 1.0$, $\rho = 0.5$, $\sigma_U = 0.01$ and $\sigma_Q = 0.1$. The same parametric shape for λ was used for both discrete and continuous formulations. Incidence data are shown in Figure 7.

Results Formulation (13) was implemented in R/JAGS (uses classical MCMC[10]) and the continuous version (16) with R/Stan (uses Hamiltonian MCMC coupled with NUTS algorithm). Results for the estimation of the parametric-shaped generation interval distribution are shown in Table 1 and for the non-parametric one in Figure 8.

Table 1: *Estimation of parametric generation interval distribution. Mean and 95% CI.*

	True	MCMC	H-MCMC
a	1.00	1.50 (1.44-1.63)	0.98 (0.89-1.08)
$\mathcal{R}_0$	1.80	1.25 (1.10-1.67)	1.82 (1.73-1.92)

Important Disclaimer: More work would have been needed to fine-tune JAGS (and to a lesser extent Stan) but I unfortunately did not have the time to finalize this within the time allowed for this comprehensive examination. Consider results and interpretations as illustrative.

Interpretation The Hamiltonian MCMC manages to give a relatively good estimation of the parametric generation interval distribution. Generally, estimating both $\mathcal{R}_0$ and the generation interval distribution typically raises an issue of identifiability between these two parameters [12], here the parametric shape certainly helps by reducing the dimension of the problem (but at the price of a strong assumption on the distribution shape). It is surprising that a better estimation could not be reached with JAGS given the low dimensionality (there were no obvious convergence issues).

The estimation of a non-parametric distribution is a much harder problem because the formulation specified $T = 10$ independent variables (which are actually not so independent). But again, Hamiltonian MCMC manages relatively well to estimate the general shape of the distribution (bottom panel Figure 8) whereas MCMC with a standard sampler did not manage to successfully estimate it (top panel of Figure 8).

Finally, the simulated data set were different for the discrete and continuous formulations. Maybe the continuous one was, by chance, easier to fit than the discrete one. To test this, the fitting exercise should be performed several time on multiple simulated data. I did not have the time to fully investigate this.

3.2 Scientific hypothesis testing

Aside from process and observation errors, "model design" error is another potential issue when modelling an epidemiological process that is not fully understood. This type of error stems from a modelling assumption that is not consistent with the reality of the system considered. Models are always a simplification of reality and modellers have always been aware of the danger of incorrect model design. But only recently in the epidemiological modelling literature, new statistical tools combined with increased computing power and data collection have enable them to test model design rigorously [24, 17, 7, 5]. Indeed, if there are several modelling design candidates for a given observed data set, selection can be performed based on statistical inference. Here, statistical methods are not used in the sole goal to infer the parameter values that best explain the data from *one* model. Instead, these methods can help select the model design that best explains the observed data by comparing their respective relevant statistics.

Model selection Many statistical tools have been developed to guide model selection and reviewing them all is beyond the scope of this report. The key idea across all these methods is to use statistical quantities (e.g. likelihood, acceptance ratio, etc.) derived from inferring the best parameters for each model and then rank candidate models according to a measure of their ability of fitting the observed data. For this study, I arbitrarily chose the Akaike Information Criteria (AIC) [**?**] and Bayesian factors (BF) [16]. The AIC is simply defined as

$$AIC = 2(\, p - \log(L^*)\,)$$

where p is the number of model parameters fitted to data and L^* is the associated maximum likelihood. Models with the smaller AIC are preferred.

Bayesian factors compare models pairwise. Given two models A and B with $p(\theta_A)$ and $p(\theta_B)$ as priors on their parameters and $p(\theta_A|D)$ and $p(\theta_B|D)$ their posterior distributions, the bayesian factor of A versus B is defined as

$$BF_{AB} = \frac{p(\theta_A|D)\,/\,p(\theta_B|D)}{p(\theta_A)\,/\,p(\theta_B)}$$

The larger is BF_{AB} the more evidence we have in favour of model A. A table of (arbitrary) values associated with a practical interpretation commonly used is given in Table 2.

Table 2: *Bayesian factors interpretation*

$\mathbf{BF_{AB}}$	**Evidence in favour of** A
1 - 3	very weak
3 - 20	positive
20 - 150	strong
> 150	very strong

Epidemiological Models As a concrete application, let's consider an epidemiological model of Ebola virus. The recent outbreak in West Africa has triggered a wave of modelling studies to investigate a wide range of outcomes (e.g., incidence forecast, risk of exporting cases outside outbreak regions, treatment prioritization, etc.). Some studies made different model design choices for addressing the same question [1, 11]. It

would then be insightful to assess the various modelling assumptions with rigorous statistical tools.

Reference model In order to test the statistical methods efficiency, a reference ("true") model is designed and incidence data are simulated from it. The population is segregated into six compartments: susceptible individuals (S), exposed individuals (E) who have been infected but are not yet infectious, infectious individuals who are both infectious (I) and symptomatic, individuals who are infectious but asymptomatic (A). Asymptomatic infections occur with probability ρ. The disease induces a specific mortality rate f. Infected individuals who die undergo a funeral ceremony (F) where contact with the corpse (by relatives attending the ceremony) can still transmit the disease. Individuals who cannot transmit anymore (i.e. recovered or buried) are moved to compartment R. Lifelong immunity is assumed. The relative infectiousness of individuals in compartments A and F compared to the ones in I is represented with multiplicative factors α_A and α_F respectively. The average length of residency in a compartment x is D_x. This reference model is labelled SEAIFR. Its implementation is stochastic and discrete both in time and in population units (Gillespie). It is summarized in Table 3 and depicted in Figure 9 (top panel).

The data consist of one realization of SEAIFR model with parameters given in Table 4.

Table 3: *Stochastic implementation of the SEAIFR model*

Event	Demographics	Rate
Infection	$(S, E) \to (S-1, E+1)$	$S\beta_I(I + \alpha_A A + \alpha_F F)$
Clinical onset	$(E, I) \to (E-1, I+1)$	$((1-\rho)/D_E)E$
Sub-clinical	$(E, A) \to (E-1, A+1)$	$(\rho/D_E)E$
Asympt. recovery	$(A, R) \to (A-1, R+1)$	A/D_A
Sympt. recovery	$(I, R) \to (I-1, R+1)$	$(1-f)I/D_I$
Sympt. death	$(I, F) \to (I-1, F+1)$	fI/D_I
Corpse buried	$(F, R) \to (F-1, R+1)$	F/D_F

Table 4: *Parameters for the simulated incidence*

Parameter	Value
$\mathcal{R}_0$	1.8
D_L	5 days
D_I	4 days
D_A	3 days
D_F	4 days
α_A	0.2
α_F	4.0
ρ	0.15
f	0.4
reporting rate	0.25
pop. size	10,000

Candidate Models Now, let's assume the disease transmission dynamics of the observed outbreak are not well understood. There are two epidemiological hypothesis:

- H_a: asymptomatic infections occur
- H_f: transmission occur during funeral

ceremonies following disease-induced death

Four candidate models are proposed to test these two hypothesis

- SEAIFR model: consider both H_a and H_f hypothesis
- SEIFR model: ignores asymptomatic infections but consider the possibility of transmission during funerals.
- SEIAR model: ignores transmission during funerals but consider the possibility of asymptomatic infections.
- SEIR: ignores both H_a and H_f hypothesis

The first candidate model is the reference model, so naively we could expect this model to perform well when compared to others.

In order to test the validity of hypothesis H_a and H_f, statistical inference is performed on the four models. I used IF1 and IF2 to derive the maximum likelihood. The Bayesian factors were derived from an Approximate Bayesian Computation (ABC). The principle of ABC is simple: *i)* parameters to be fitted are drawn from a prior distribution *ii)* the model is run with those parameters *iii)* the distance between the data and the relevant model output (e.g., incidence) is calculated *iv)* if this distance is smaller than an arbitrary threshold the parameter drawn at the first step is accepted as part of the posterior distribution. Here I chose a distance metric that is the squared difference between the smoothed incidences after their respective peaks have been aligned.

Important Disclaimer: **Parameters and likelihood estimation with IF1 and IF2 from the R package `pomp` did not lead to sensible results. More work would have been needed to fine-tune but I unfortunately did not have the time to finalize this within the time allowed for this comprehensive examination. Consider results and interpretations from IF1/2 as illustrative.**

Results Table 5 shows **hypothetical** results of the AIC for the four models with their likelihood calculated from IF1 and IF2 and Table 6 shows calculated Bayesian factors between pairs of candidate models.

Table 5: ***Hypothetical*** *results from IF1 and IF2*

model	n	loglik IF1	loglik IF2	AIC_1	AIC_2
SEAIFR	8	-249	-248	514	512
SEAIR	6	-256	-250	524	512
SEIFR	5	-255	-250	520	510
SEIR	3	-256	-255	518	516
SIR	2	-258	-257	520	518

Table 6: *Bayesian factors from ABC*

models	**BF**
SEAIFR vs SEIFR	0.90
SEAIFR vs SEAIR	1.52
SEAIFR vs SEIR	18.71
SEAIFR vs SIR	14.81
SEIFR vs SEAIR	1.69
SEIFR vs SEIR	20.79
SEIFR vs SIR	16.46
SEAIR vs SEIR	12.32
SEAIR vs SIR	9.75
SEIR vs SIR	0.79

Interpretation (Bayesian factors only) The "real" model was designed such that the funeral compartment is important: its force of infection is relatively large ($\alpha_F = 4$) and occurs with a time lag. Hence we would expect models with an F compartment to be selected for versus one model which does not have it. This is what we see, but only for pairs of simple models (e.g., SEIFR vs SEIR). For models with more compartments, this simulation could not find a strong evidence for including funerals in the epidemiological modelling.

4 Discussion

Summary

Mathematical models are invaluable to help understand complex real-life systems. These models often involve parameters whose values are not *a priori* known. Because the system being modelled is observable, the natural way is to "reverse-engineer": what are the parameter values that make the model behaves like the observed system? When models involve parameters and variables that are either not observable or difficult to measure accurately enough, reverse-engineering can be challenging. Add a large number of parameters to estimate and the task can be impossible. Incorporating a statistical modelling layer focuses parameters estimation into maximizing a likelihood (if we believe parameters have a fixed value) or calculating an expected value (if we believe parameters are random variables themselves). But here again, if "naive" methods are used, the task of maximizing a likelihood or calculating an expectation may quickly become impossible (even with the best computing power available) due to the complexity and/or dimensionality of the model.

For specific frameworks, the statistical methods reviewed here provide techniques that can improve dramatically parameter estimation efficiency. Furthermore, these methods can be applied to models without making strong linearity assumptions about the system modelled, as was typically the case before the time when only analytical methods were available (late 1980s).

Particle filters are well adapted when the system is modelled as a memoryless (Markovian) and partially observed process (POMP), thanks to their recursive formulation (using current data information to infer unobservable variables at the next time step). However, when the dimension increases, the number of particles must increase exponentially because the parameter space needs to be covered appropriately. Failure to do so can result in a poorly approximated posterior distribution where some particles are disproportionately weighted.

Iterated filters like IF1 and IF2 take the basic idea of the particle filter and add a layer of iterations to locate more easily the maximum likelihood. Successive iterations "erode" the likelihood surface in a similar way data cloning does. But unlike basic data-cloning, IF1 and IF2 use inference at the end of each iteration to be used into the next one. Although IF1 and IF2 are both iterated filters, their inner mechanics are different: IF1 converges to a single maximum likelihood estimate based on interim statistics of the underlying particle filter. IF2 use perturbations while performing the particle filter and its output is a set of particles (not a single value) converging to the maximum likelihood point. Because they are underpinned by particle filters, both IF1 and IF2 are restricted to POMP.

When model parameters are assumed stochastic, their inference boils down to calculating an expectation, that is an integral with respect to the posterior density $p(\theta|D)$ which is typically impossible to sample from and/or high dimensional, precluding basic Monte Carlo techniques for the former case and grid-integration for the latter. Markov chain Monte Carlo essentially constructs a Markov Chain that converges to the posterior density of interest such that standard Monte Carlo integration can be performed (using the Markov chain values as the integrating samples). Speed of convergence of the Markov chain depends on the algorithm that constructs it. Hence a lot of efforts has been put into designing efficient samplers, but efficiency can be model dependent. Correlation between parameters and high dimensionality typically hinder

convergence.

Introducing Hamiltonian dynamics into MCMC can significantly improve both speed of convergence and exploration of the parameter space. Unlike other MCMC sampling algorithms like Metropolis-Hasting which have more or less a random walk behaviour, Hamiltonian MCMC calculates the gradient of the target density giving information about its topology, hence an efficient exploration. Because of the gradient calculation, Hamiltonian MCMC is restricted to continuous distributions, which can be problematic in some applications.

MCMC algorithms require that the target density can be explicitly evaluated, which can be costly or even impossible in some cases. As it was reviewed with exact-approximate MCMC, this limitation can be lifted if the target density can be numerically approximated in a unbiased fashion. So, the range of application of MCMC can be expanded further, but maybe at the price of slower convergence (as illustrated in Figure 6 with a toy example).

Applications

I attempted to use the statistical techniques reviewed above to epidemiological applications.

The first application aimed at estimating the generation interval distribution from partially observed incidence data. The mathematical model describing the epidemiological process was not Markovian, precluding the use of particle filter based methods. First, no assumption was made about the shape of the generation interval distribution (non-parametric estimation). Hence, each day of the age of infection was an independent parameter to estimate defining a high-dimensional (dimension=10), correlated estimation problem. Estimation with Hamiltonian MCMC was relatively good compared to a more traditional MCMC algorithm which failed to give a satisfactory estimation for the generation interval distribution. Indeed, Hamiltonian MCMC is expected to perform better in high-dimension and correlated problems. Then, the dimension of the parameter space was reduced to 2 by assuming a parametric shape for the generation interval distribution. Hamiltonian MCMC manage once again to successfully estimate the true (simulated) values of the parameters of interest. However, I would have expected classical MCMC to perform better in this parametric case, but it is likely a lack of fine-tuning of the software (JAGS) from my part that prevented a better estimation.

The second application investigated model selection. Indeed, depending of the statistical method used, by-product of parameters estimation can be used to mount evidence for or against different candidate hypothesis to include in mathematical models. This evidence is based only on observed data from the system modelled. Here I used both iterated filters IF1 and IF2 and approximate bayesian computation (ABC) to test, from partially observed incidence data only, if asymptomatic infections and transmission during funerals for an Ebola-like disease were hypothesis supported by data. The by-product from IF was the likelihood which was used for the Akaike Information Criteria (AIC), whereas bayesian factors were calculated from ABC simulations. I only managed to proceed with ABC, and failed to get sensible estimations from IF1/2 (again, I probably did not fine-tune the software (pomp) well enough). Because model selection is a by-product from parameter estimation, the same kind of issues can occur. Identifiability issues with parameters translates in neutral evidence between two mathematical models despite being conceptually different. The results I had for the model selection reflected this problem: despite the "true" model being

constructed such that the funerals compartment plays an important role, it was not systematically identified as a key component of the epidemiological process. More generally, different parts of the data can be explained by different model designs [24].

Limitations

I segregate the limitations of the statistical methods reviewed here in two categories: theoretical and practical.

On the theoretical side, these methods are limited to a specific framework. Particles filter based methods operate within a Markovian framework, hence limiting their application to "memoryless" mathematical models. This limitation was illustrated with the choice of the first application topic that used the simple, yet epidemiologically fundamental, renewal equation which is not Markovian. Also, high dimensionality can be an issue because the number of particles needed doesn't scale conveniently with dimension and sample impoverishment is more acute[23]. In IF1/2, the flip side of contrasting the maximum likelihood thanks to the iterations may be that the flattening of other regions of the likelihood increases the risk that particles explore irrelevant values for a long time.

MCMC methods do not require a Markov property. However, unlike particles based methods, they need to evaluate explicitly the target distribution or, in the case of exact-approximate MCMC, an unbiased approximation of it. This can be an issue for some applications. For example, parameters of a moderately complex epidemiological compartmental model cannot use MCMC inference for its parameters because the posterior density $p(\theta|D)$ is intractable. High dimensionality may be an issue because of the increasing difficulty to build a Markov chain that efficiently explore the parameter space. Correlation between parameters makes this problem worse. Hamiltonian MCMC is a sampler of choice for such situations, but is restricted to continuous target distributions.

I briefly mention approximate bayesian computation (ABC) as it was used in one of the applications. ABC can be very appealing because it does not impose any theoretical constraint on the underlying mathematical model (e.g., no Markovian behaviour, can be as complex as desired). However, defining the rejection distance may involve decisions too arbitrary and the statistical power of ABC, at least in its simplest form, is very weak. For the model selection application, acceptance rate was less than 1%.

The practical limitations are, for all methods reviewed here, diagnostics and computation time. Indeed, as the model complexity or dimensionality increases, it is very difficult to ascertain when either method has reached its goal (e.g., maximum likelihood, convergence to a probability distribution). Similarly, months or even years of CPU time are not uncommon for practical problems. However, this will likely be partially solved with increasing hardware performance.

Despite all their respective limitations, this limited set of statistical methods constitute a relatively flexible toolbox that covers a broad range of applications.

Acknowledgment: Darren Wilkinson's research blog (https://darrenjw.wordpress.com) was very useful.

5 Figures

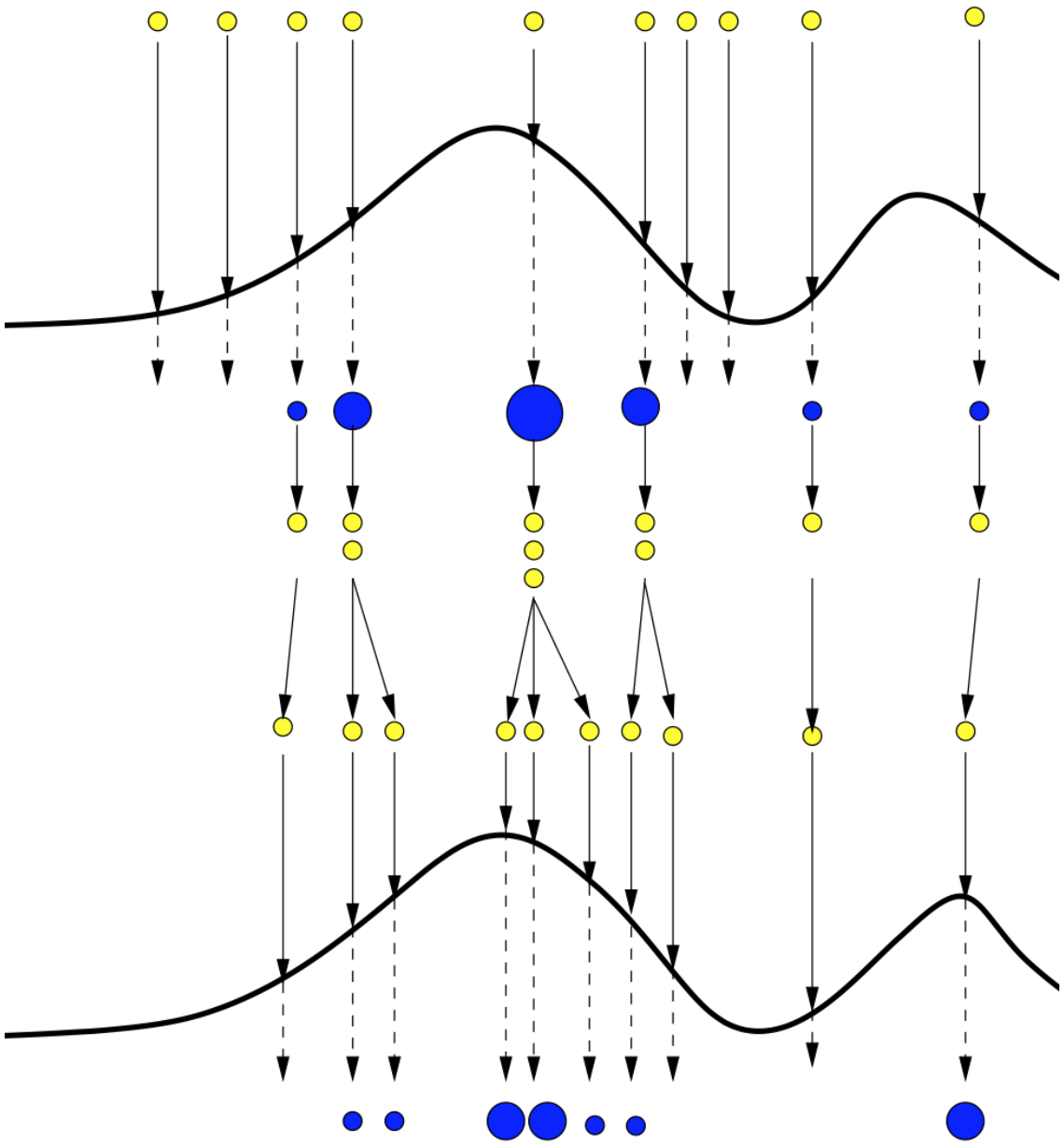

Figure 1: Illustrative example of Particle Filtering algorithm *(figure taken from [6]). The first line of yellow points represent the initial ($t = 0$) sampling of $N = 10$ particles $x^1, .., x^{10}$ from distribution p_0. The likelihood of the observation distribution $p(y_{t=0}|x^1, ..., x^{10})$ is used to calculate the weights (blue circles). The larger the likelihood at a given sample, the larger the associated weight is. The second line of yellow points shows the new set of particles $\tilde{x}^1, .., \tilde{x}^{10}$ after being resampled, with replacement, according to the weights. The third line of yellow points shows how the particles are propagated using the dynamical process f_1. Then, these particles are used as the starting point for the next time step.*

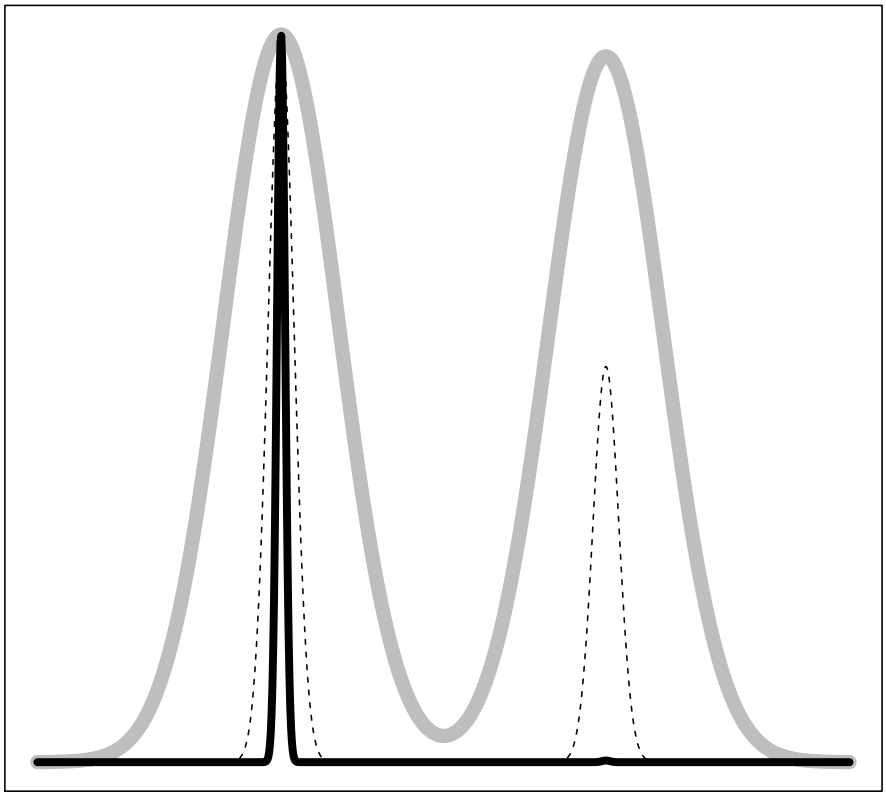

Figure 2: Data cloning effect on likelihood. *Artificially cloning observation data has the effect of exponentiating the likelihood and hence facilitating the identification of a global maximum. The grey curve represents a fictive initial likelihood L. Note the local maximum (right) has a value very close to the global one, making it difficult for an algorithm to find the global one once it has found this local. The dash curve shows L^{20}: the local maximum has been significantly reduced. The black curve shows L^{200} where the global maximum stands-out.*

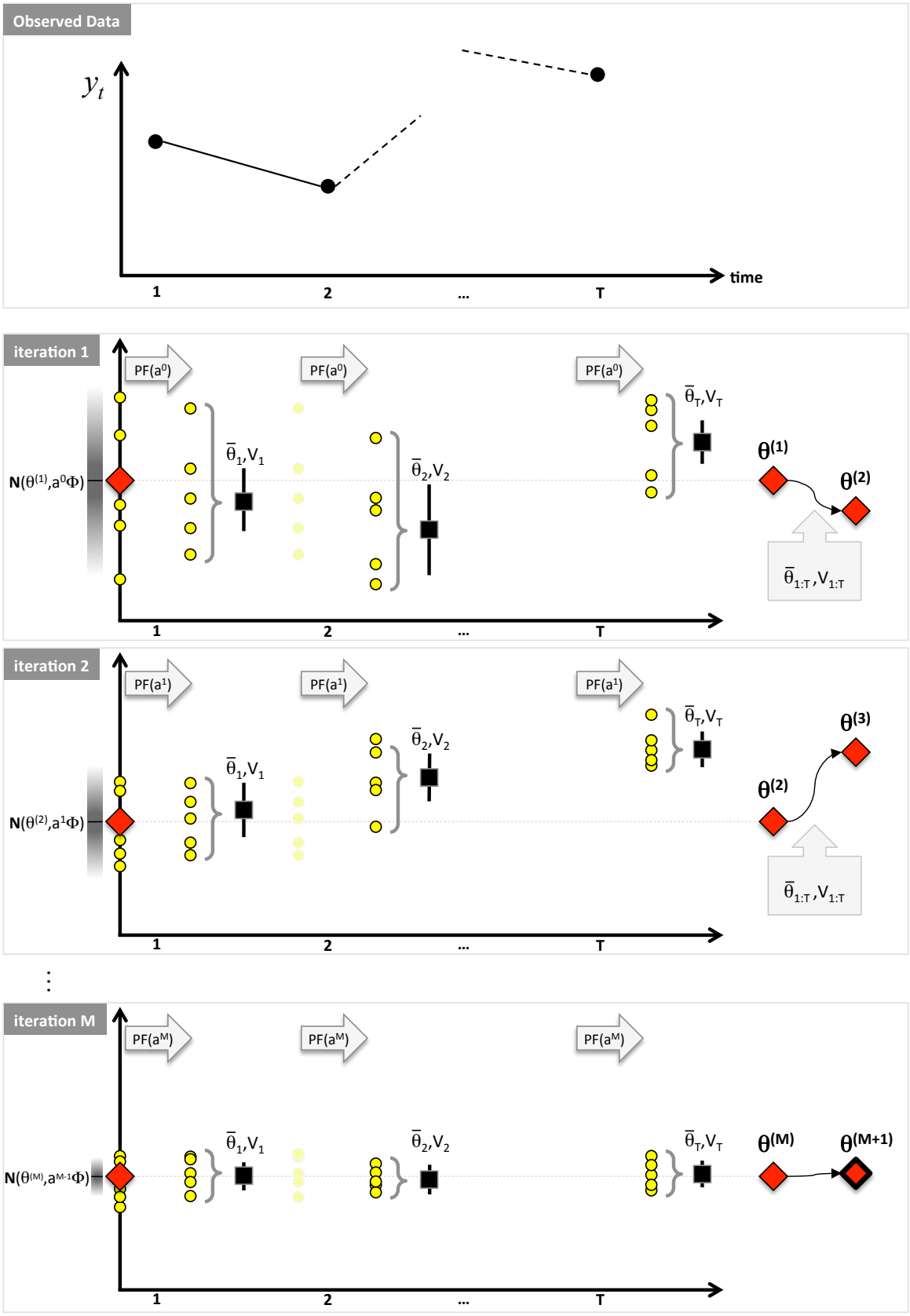

Figure 3: Illustrative example of Iterative Filtering 1 *[14]. See main text for explanations.*

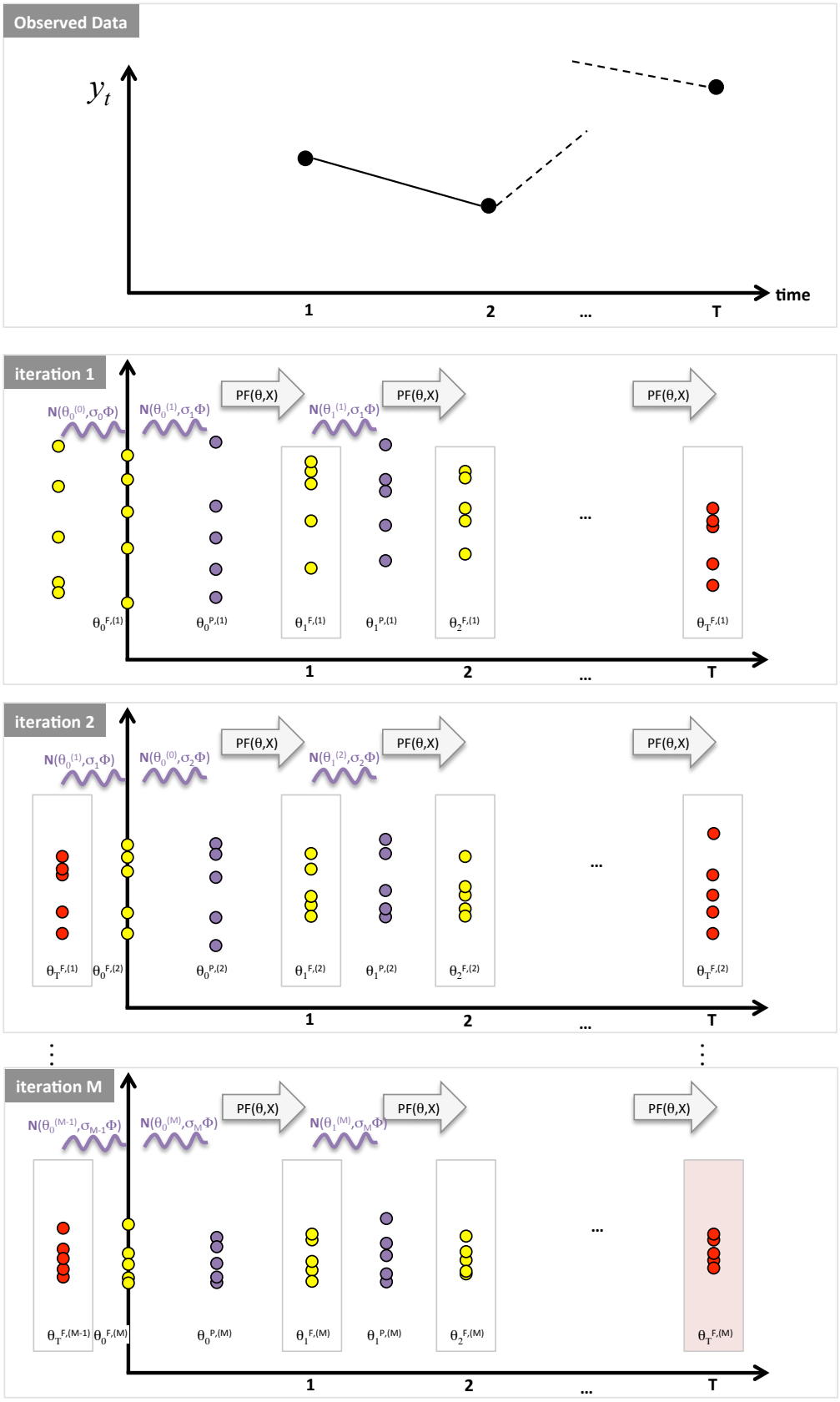

Figure 4: Illustrative example of Iterative Filtering 2 [15]. *See main text for explanations.*

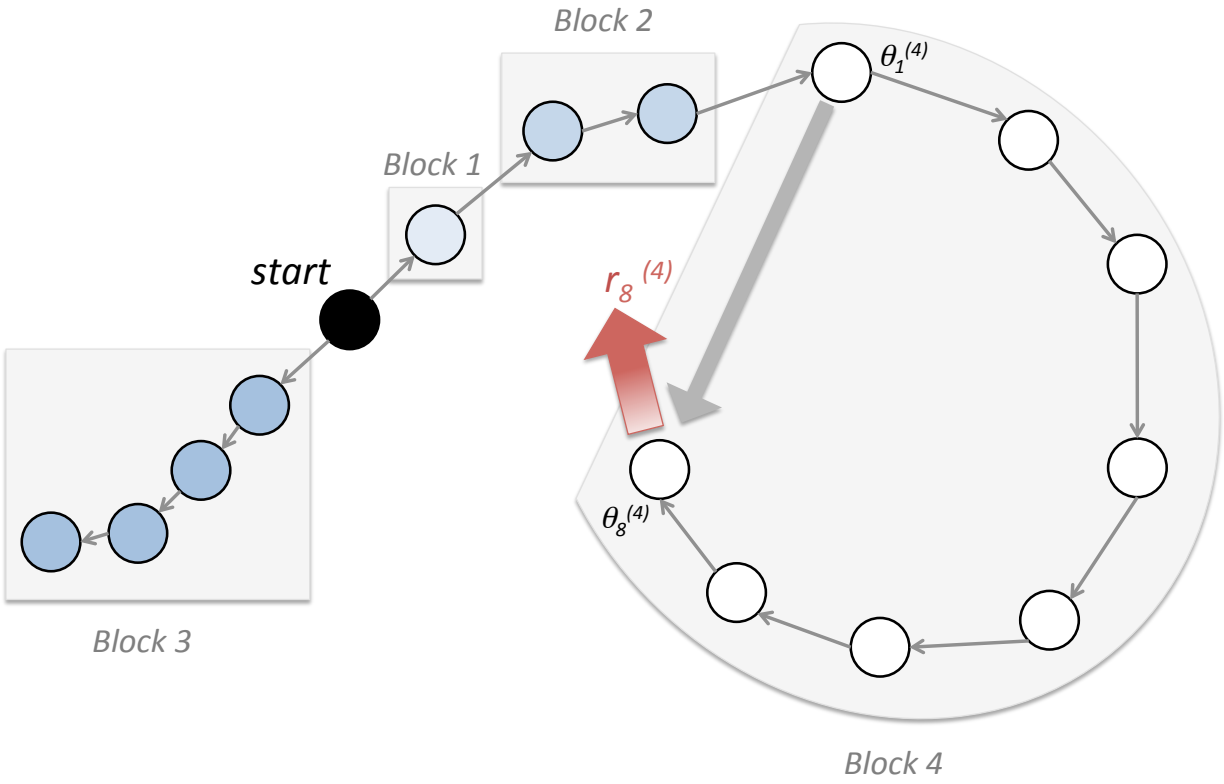

Figure 5: Illustrative example of NUTS algorithm. *The algorithm starts from a previously accepted sample (solid black point). The first block is randomly chosen to leapfrog-integrate forward in time, for 1 (=2^0) step only. No stopping conditions were triggered at that point. Time direction is randomly chosen for a second time and happens to be forward again. Hence a second trajectory block made of 2 (=2^1) steps is constructed from the last position which was integrated forward, and again no stopping condition triggered. The time direction of the third block is randomly drawn backward and 4 (=2^2) steps are integrated from the last position backward-integrated, which is in our case the starting point. The fourth block is chosen to integrate forward in time and will generate 8 (=2^3) new positions. However, a stopping condition is triggered: the trajectory starts to make a U-turn because the angle between the vector (thick dark grey arrow) defined by the starting position of this fourth trajectory (labeled $\theta_1^{(4)}$) and the last one (labeled $\theta_8^{(4)}$) makes an angle wider than 90 degrees with the moment vector at that last position ($r_8^{(4)}$ thick red arrow), that is $(\theta_8^{(4)} - \theta_1^{(4)}).r_8^{(4)} < 0$. The fourth block is ignored and the pool of candidate samples consists of all positions from blocks 1,2 and 3 (blue circles). The next Hamiltonian MCMC sample proposal will be randomly (uniformly) selected from this pool.*

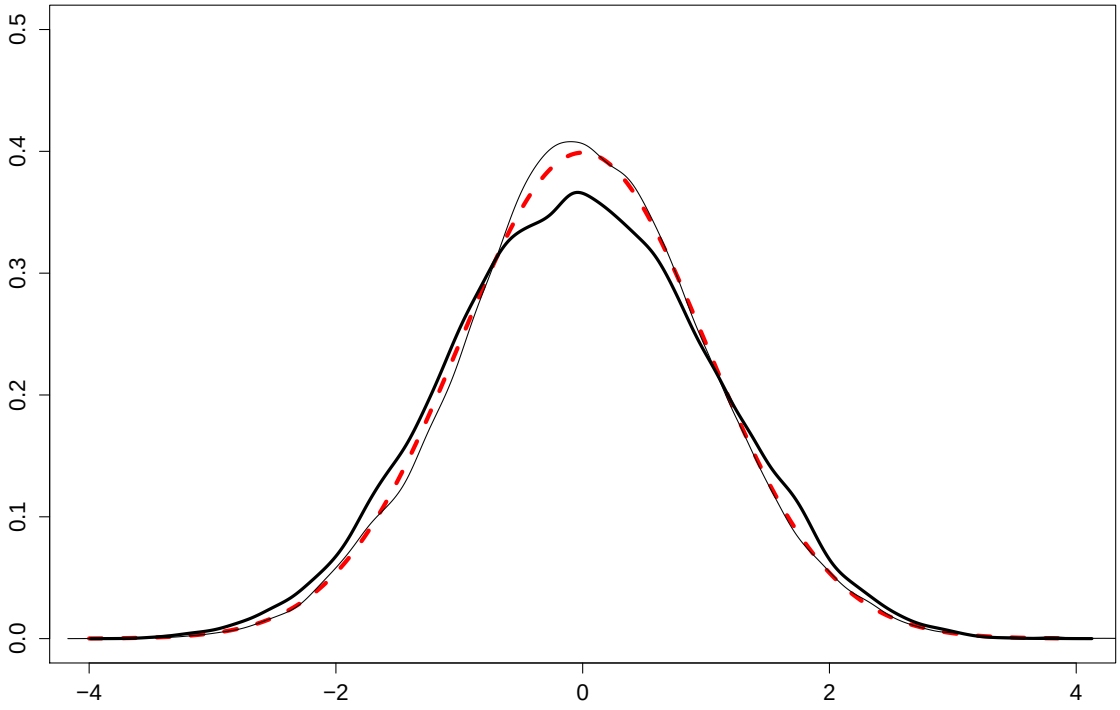

Figure 6: MCMC using approximated target distribution. *The target distribution (dash red) is a gaussian $N(0, 1)$. One MCMC run was used with the exact evaluation of the target distribution density (thin black) and another one (thick black) with artificial noise added to its evaluation, mimicking estimation error (say, from particle filter). Convergence was slower for the perturbed distribution. Chain iterations: 50,000 (after 50,000 warm-up)*

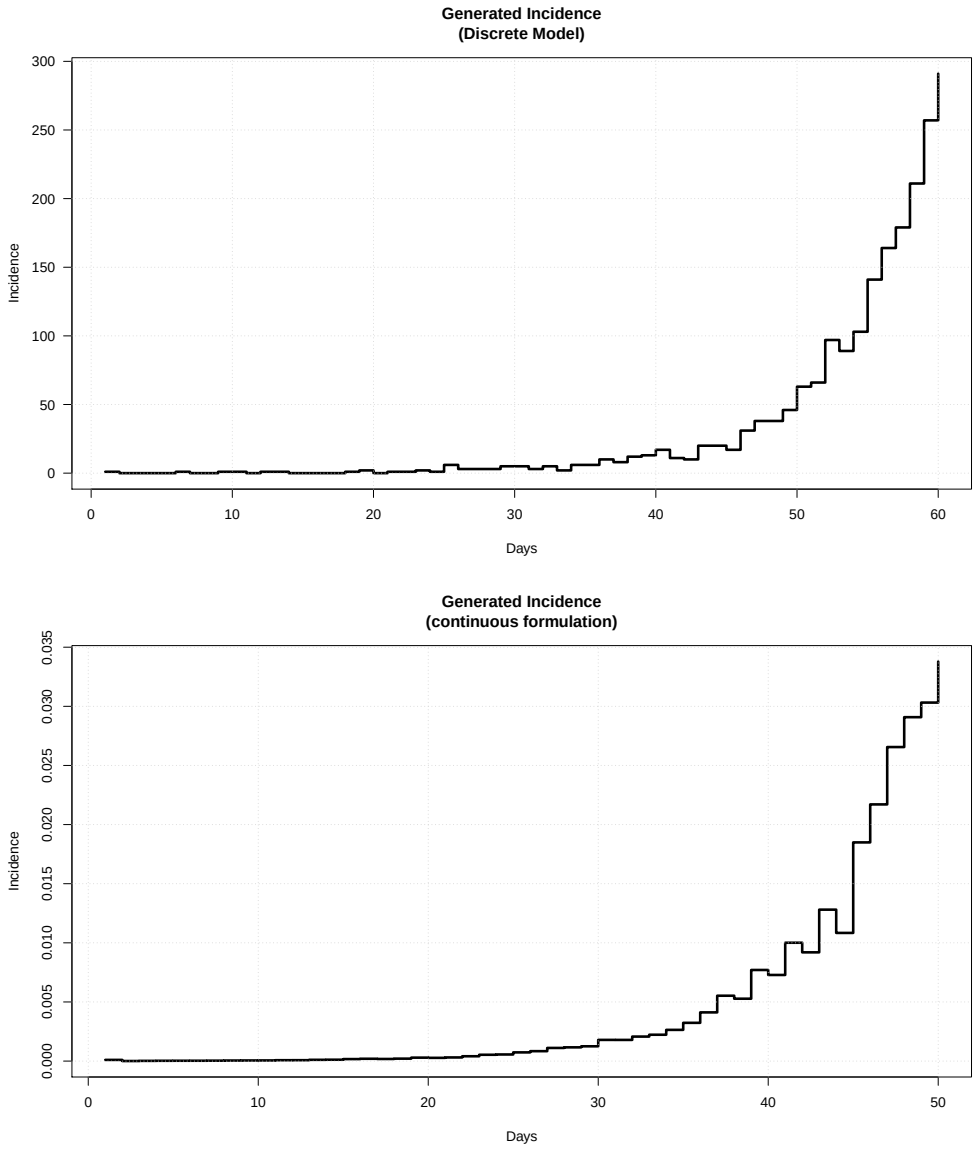

Figure 7: Simulated incidence for generation interval estimation *The top panel shows incidence generated with the discrete model (13) and the bottom panel with the continuous one (16).*

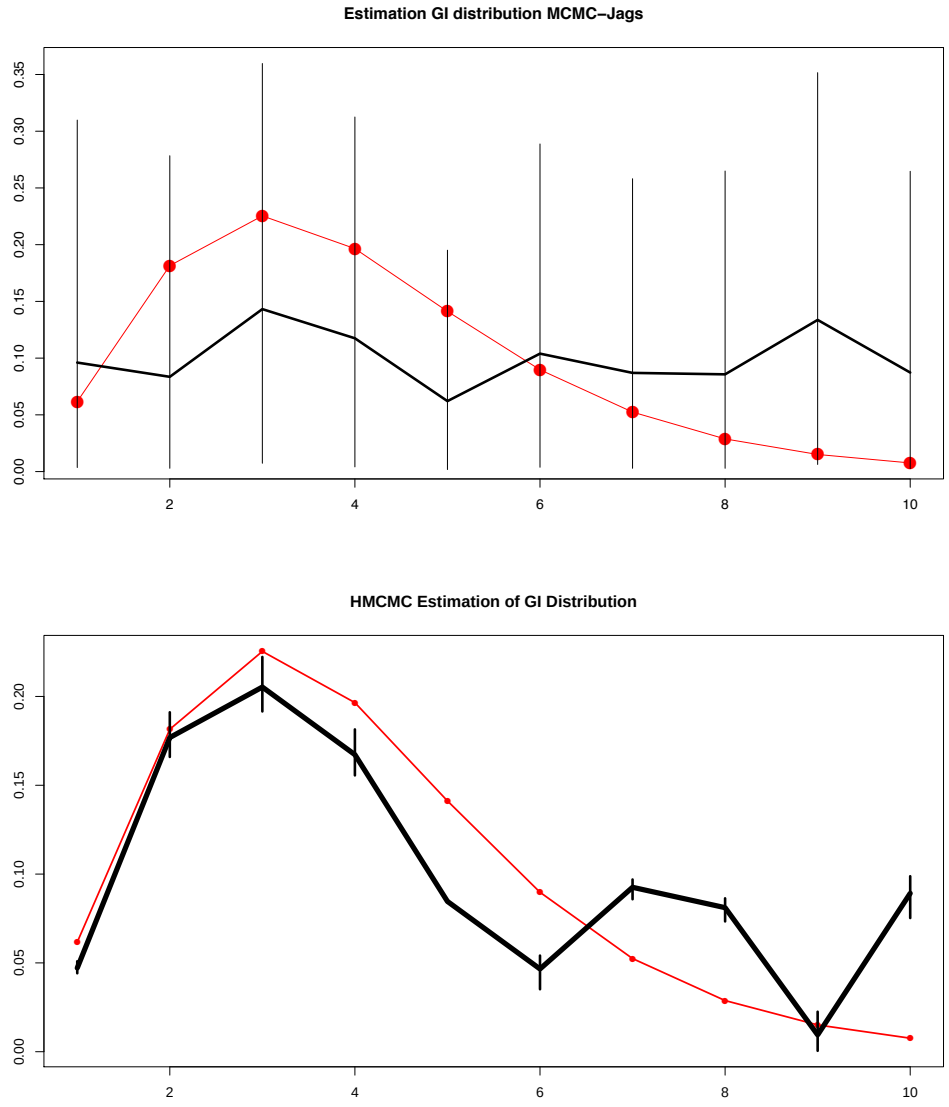

Figure 8: Estimation of non-parametric generation interval *The red thin curve represents the "true" generation interval distribution used to simulate incidence data. The black thick line shows the estimated mean and the vertical segments the 95% credible intervals. Top panel: standard MCMC estimation (R/JAGS) with the discrete formulation (13). Bottom panel: Hamiltonian MCMC coupled with NUTS algorithm (R/Stan) with the continuous formulation (16).*

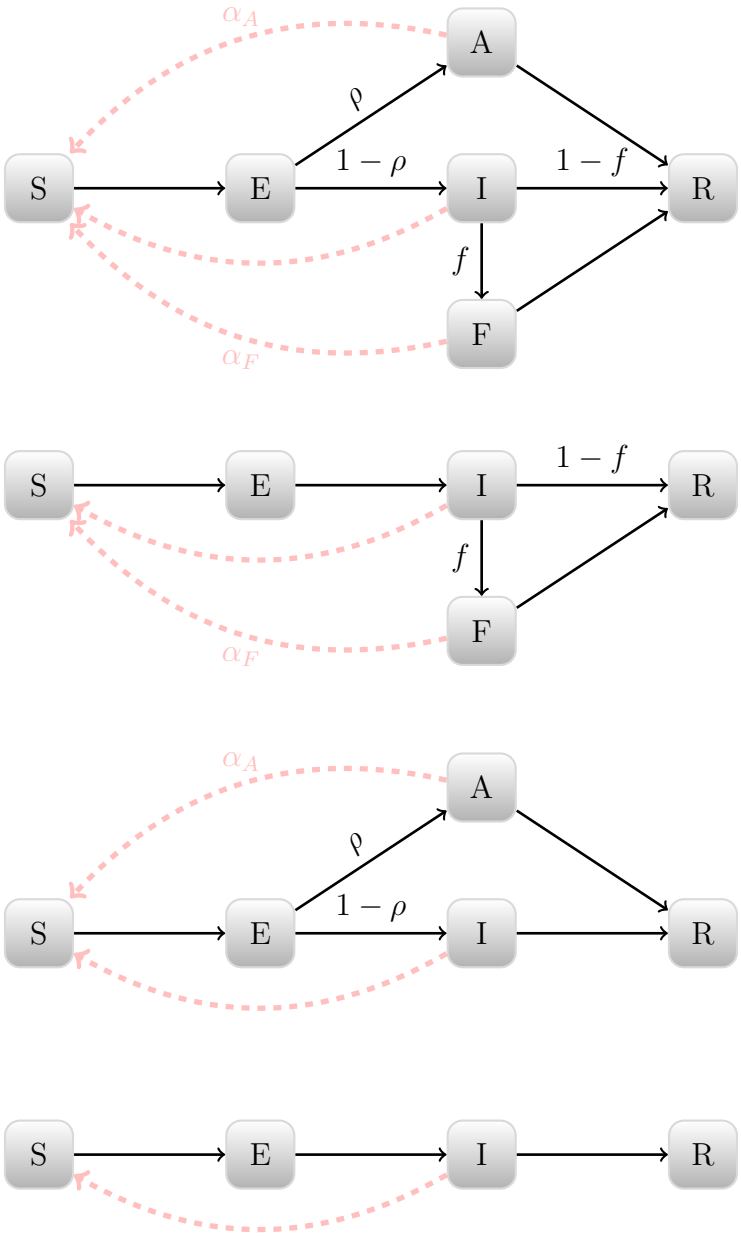

Figure 9: *Diagram of the candidate models for hypothesis testing. The top panel represents the SEAIFR model which generated the data and is also part of the candidate models. In the second panel, the candidate model SEIFR ignores asymptomatic infections. The SEAIR model in the third panel ignores infections during funerals. Finally, a SEIR model is represented in the bottom panel and ignores both asymptomatic and funeral-induced infections.*

A Algorithms

Algorithm 1: Particle Filtering.

Data: Observation time series $y_1, ..., y_T$

Input: number of particles N, initial sampling distribution p_0

/* Initialization */

Sample N particles from p_0: $x^1, ..., x^N$

for $(t = 1, 2, ..., T)$ **do**

  /* Weights from likelihood (measurement update) */

  for $(i = 1, 2, ..., N)$ **do**

    $w^i \leftarrow p(y_t|x^i) = f_2(x^i, \theta)$

  end

  /* Normalize weights */

  for $(i = 1, 2, ..., N)$ **do**

    $w^i \leftarrow w^i / \sum_k w^k$

  end

  /* Resample according to weights */

  $\tilde{x}^{1:N} \leftarrow \text{sample}(x^{1:N}, \text{prob} = w, \text{replace} = true)$

  /* Update next time step X from the resampled particles: predicts the new particles */

  for $(i = 1, 2, ..., N)$ **do**

    $x^i \leftarrow f_1(\tilde{x}^i, \theta)$

  end

end

Result: A set of N samples approximating the continuous posterior distribution $p(x_t|y_t)$ at every time step $t = 1, ..., T$.

Algorithm 2: Iterative Filtering 1.

Data: Observation time series $y_1, ..., y_T$

Input: number of main iterations M, number of particles J, initial parameter values $\theta^{(1)}$, cooling a, cooling for initial sampling b, covariance matrix Φ, function f evaluating the process X at the next time step, observation process $g(y|X, \theta)$

for $m = 1, 2, ..., M$ **do**
  for $j = 1, ..., J$ **do**
    /* Process starting position */
    $X_F(0, j) = X(0)$
    /* Parameters initialization */
    $\theta(0, j) \sim \mathcal{N}(\theta^{(m)}, ba^{m-1}\Phi))$
  end
  /* Loop through all observation dates */
  for $t = 1, 2, ..., T$ **do**
    /* Particle filter from t-1 to t: */
    for $j = 1, ..., J$ **do**
      /* Prediction based on current knowledge: */
      $X_P(t, j) \leftarrow f(X_F(t-1, j), \theta(t, j))$
      /* weights based on likelihood */
      $w(t, j) \leftarrow g(y_t|X_P(t, j), \theta(t, j))$
    end
    /* Resampling with replacement */
    $k_1, ..., k_J \leftarrow \texttt{sample}(1 : J, prob = w(t, \bullet), replace = true)$
    /* Update parameter and process */
    for $j = 1, ..., J$ **do**
      $X_F(t, j) \leftarrow X_P(t, k_j)$
      $\theta_s(t, j) \leftarrow \theta(t, k_j)$
    end
    /* Update parameter values */
    for $j = 1, ..., J$ **do**
      $\theta(t, j) \sim \mathcal{N}(\theta_s(j), a^{m-1}\Phi))$
    end
    /* ((End particle filter)) */
    /* Mean and variance (component-wise) */
    for $d = 1, ..., \dim(\theta)$ **do**
      $\bar{\theta}_d(t) \leftarrow \text{mean}_j(\theta_d(t-1, \bullet))$
      $V_d(t) \leftarrow \text{variance}_j(\theta_d(t-1, \bullet))$
    end
  end
  /* Update parameter iterated filtered values */
  for $d = 1, ..., \dim(\theta)$ **do**
    $\theta_d^{(m+1)} \leftarrow \theta_d^{(m)} + \sum_{t=1}^{T} \frac{V_d(1)}{V_d(t)}(\bar{\theta}_d(t) - \bar{\theta}_d(t-1))$
  end
end

Result: Estimate of the parameter values $\theta^{(M+1)}$ maximizing the likelihood

Algorithm 3: Iterative Filtering 2.

Data: Observation time series $y_1, ..., y_T$

Input: number of main iterations M, number of particles J, J initial parameter particles $\Theta_j^{(0)}$, perturbation density $h(mean, var)$, perturbation sequence matrix $\sigma_{1:T}$, function f evaluating the process X at the next time step, observation process $g(y|X, \theta)$

for $m = 1, 2, ..., M$ **do**
  for $j = 1, ..., J$ **do**
    /* Process starting position */
    $\theta_F^{(m)}(0, j) \sim h(\Theta^{(m-1)_j}, \sigma_m)$
    $X_F(0, j) = f(X(0), \theta_F^{(m)}(0, j))$
  end
  /* Loop through all observation dates */
  for $t = 1, 2, ..., T$ **do**
    /* Particle filter from t-1 to t: */
    for $j = 1, ..., J$ **do**
      /* Perturbation of parameters: */
      $\theta_P^{(m)}(t, j) \sim h(\theta_F^{(m)}(t-1, j)), \sigma_m)$
      /* Prediction based on perturbation: */
      $X_P(t, j) \leftarrow f(X_F(t-1, j), \theta_P^{(m)}(t, j))$
      /* weights from perturbed likelihood */
      $w(t, j) \leftarrow g(y_t|X_P(t, j), \theta_P^{(m)}(t, j))$
    end
    $k_1, ..., k_J \leftarrow$ `sample`$(1 : J, prob = w(t, \bullet), replace = true)$
    /* Update parameter and process */
    for $j = 1, ..., J$ **do**
      $\theta_F^{(m)}(t, j) \leftarrow \theta_P^{(m)}(t, k_j)$
      $X_F(t, j) \leftarrow X_P(t, k_j)$
    end
    /* ((End particle filter)) */
  end
  /* Set all particles for next main iteration */
  $\Theta^{(m)_j} \leftarrow \theta_F^{(m)}(t, j)$ for all $j = 1, ..., J$
end

Result: Set of J parameter particles $\Theta_{1:J}^{(M)}$ located near the maximum likelihood

Algorithm 4: Metropolis-Hasting MCMC

Input: Number of samples n

/* Initialization (random or not) */

X_0 initialized

for $(t = 1, 2, ..., n)$ **do**

  /* Draw a candidate from q */

  $y \leftarrow q(.|X_{t-1})$

  /* Acceptance probability */

  $\alpha \leftarrow \frac{p(y|D)q(X_{t-1}|y)}{p(X_{t-1}|D)q(y|X_{t-1})}$

  /* Update Markov chain value */

  $u \leftarrow \text{Unif}(0, 1)$

  if $u < \alpha$ **then**

    $X_t = y$

  else

    $X_t = X_{t-1}$

  end

end

Algorithm 5: Hamiltonian MCMC with leapfrog

Input: Starting position $\theta(1)$, number of samples n, step size ϵ and trajectory length L

for $(t = 1, 2, ..., n)$ **do**

  /* Resample moments */

  $r(t) \sim \mathcal{N}(0, M)$

  $(\theta_0, r_0) = (\theta(t), r(t))$

  /* Leapfrog on Hamiltonian dynamics */

  $r_0 \leftarrow r_0 - \epsilon \nabla U(\theta_0)/2$

  for $(i = 1, 2, ..., L)$ **do**

    $\theta_i \leftarrow \theta_{i-1} + \epsilon M^{-1} r_{i-1}$

    $r_i \leftarrow r_{i-1} - \epsilon \nabla U(\theta_i)$

  end

  $r_L \leftarrow r_L - \epsilon \nabla U(\theta_L)/2$

  $(\theta', r') = (\theta_L, -r_L)$

  /* Metropolis-Hastings correction */

  $u \sim U[0, 1]$ $p_{acc} = \exp[H(\theta', r') - H(\theta(t), r(t))]$

  if $u < \min(1, p_{acc})$ **then**

    $\theta(t + 1) = \theta'$

  end

end

Listing 1: *Toy example of approximate-MCMC*

```
set.seed(1234)

### Number of samples for the Markov Chain
ns <- 5e4

### Noise for target eval
### (mimicks estimation of target by
###  unbiased MC)

noise <- function()
{
  return(rlnorm(1,meanlog = 0,sdlog = 0.3))
}

### Target stationary distribution: Normal(0,1)
target.mean <- 0
target.sd <- 1

target.eval <- function(x)
{
  return(dnorm(x =x,mean=target.mean,sd=target.sd)*noise())
}

### Proposal distribution ('q') parameters
### distribution = uniform[X-a; X+a]
prop.a <- 0.6

proposal.eval <- function(x,knowing)
{
  return(dunif(x=x, min=knowing-prop.a, max=knowing+prop.a))
}

proposal.draw <- function(knowing)
{
  return(runif(n=1, min=knowing-prop.a, max=knowing+prop.a))
}

### The main Markov chain
X <- vector()

### Initialize Markov chain
X[1] <- 1.234

### Acceptance rate
acc.rate = vector()

### Markov chain construction
### that should converge to the
```

```
### target stationary distribution

for(i in 2:ns)
{

  # Candidate for the new value of the markov chain
  # drawn from proposal distribution
  Y <- proposal.draw(knowing = X[i-1])

  # Acceptance probability calculation

  # 1/ density ratio of target distribution
  pi.X = target.eval(X[i-1])
  pi.Y = target.eval(Y)

  # 2/ density ratio of proposal distribution
  q.X.Y = proposal.eval(X[i-1], knowing=Y)
  q.Y.X = proposal.eval(Y, knowing=X[i-1])

  # actual probability
  proba.accept = pi.Y*q.X.Y/pi.X/q.Y.X

  # Test for acceptance
  u = runif(n=1,min=0,max=1)
  newval = X[i-1]
  acc.rate[i] = 0
  if (u<proba.accept)
  {
    newval = Y
    acc.rate[i] = 1
  }

  # update new value for markov chain
  X[i] = newval

}
```

B Examination question

Question given to candidate, 30 Mar 2015
Report due to committee, 27 Apr 2015 at noon
Oral Examination date 6 May 2005

Parameter estimation for nonlinear stochastic dynamical systems

Nonlinear stochastic dynamical systems are used to address many scientific research questions. A key question is how to use observations to estimate parameters for such systems.

You should produce an overview of state-of-the-art methods for fitting nonlinear stochastic dynamics systems to observational data, focusing on methods that account for observation error as well as underlying dynamical stochasticity (process error). You should focus on:

- Particle filtering methods, including modern versions (e.g., the IF2 algorithm)
- Markov Chain Monte Carlo methods (MCMC), including modern versions (e.g., Hamiltonian MCMC, and no-U-turn samplers (NUTS))

Other approaches (approximate Bayesian computation, optimization-based methods) may be discussed briefly, but you should not focus on these.

You should also propose two concrete areas of possible research in the applicability of these methods. Each proposal should involve:

- a scientific research question that can be studied with nonlinear stochastic dynamical systems
- discussions of obstacles or uncertainties in how parameters can efficiently be estimated for this system, and
- suggestions for how existing methods could be improved or compared for the purposes of application to the question

The proposed projects should be practical, but you do not need to solve them.

You are responsible for reviewing the relevant literature. The following links are provided only as possible starting points. http://www.pnas.org/content/112/3/719.full ; http://www.stat.columbia.edu/ gelman/research/published/nuts.pdf

The work should be done entirely by you, and the submitted document should not exceed 20 pages, in 12-point type with reasonable margins.

References

[1] Christian L Althaus. Estimating the reproduction number of Ebola virus (EBOV) during the 2014 outbreak in West Africa. *arXiv.org*, August 2014.

[2] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[3] Christophe Andrieu and Gareth O Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *Annals of statistics*, 37(2):697–725, April 2009.

[4] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.

[5] A Camacho, S Ballesteros, A L Graham, F. Carrat, O Ratmann, and B Cazelles. Explaining rapid reinfections in multiple-wave influenza outbreaks: Tristan da Cunha 1971 epidemic as a case study. *Proceedings of the Royal Society B: Biological Sciences*, 278(1725):3635–3643, November 2011.

[6] Arnaud Doucet, Nando de Freitas, and Neil Gordon. An Introduction to Sequential Monte Carlo Methods. In *Sequential Monte Carlo Methods in Practice.*, pages 1–12. Springer-Verlag, New York, November 2011.

[7] David J D Earn, D. He, M.B. Loeb, K. Fonseca, B.E. Lee, and Jonathan Dushoff. Effects of school closure on incidence of pandemic influenza in Alberta, Canada. *Annals of Internal Medicine*, 156(3):173–181, 2012.

[8] Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–511, 1992.

[9] John Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. 196, 1991.

[10] W R GILKS, N G BEST, and KKC TAN. Adaptive Rejection Metropolis Sampling Within Gibbs Sampling. *Applied Statistics-Journal of the Royal Statistical Society Series C*, 44(4):455–472, 1995.

[11] Marcelo FC Gomes, Ana Pastore y Piontti, Lucas Rossi, Dennis Chao, Ira M Longini, Jr., M Elizabeth Halloran, and Alessandro Vespignani. Assessing the International Spreading Risk Associated with the 2014 West African EbolaOutbreak. *PLoS Computational Biology*, pages 1–20, September 2014.

[12] J T Griffin, T Garske, A C Ghani, and P S Clarke. Joint estimation of the basic reproduction number and generation time parameters for infectious disease outbreaks. *Biostatistics*, 12(2):303–312, March 2011.

[13] Matthew D Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *arXiv.org*, November 2011.

[14] E L Ionides, C Bretó, and A A King. Inference for nonlinear dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 103(49):18438–18443, 2006.

[15] Edward L Ionides, Dao Nguyen, Yves Atchadé, Stilian Stoev, and Aaron A King. Inference for dynamic and latent variable models via iterated, perturbed Bayes maps. *Proceedings of the National Academy of Sciences*, 112(3):719–724, January 2015.

[16] Robert E Kass and Adrian E Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.

[17] Aaron A King, Edward L Ionides, Mercedes Pascual, and Menno J Bouma. Inapparent infections and cholera dynamics. *Nature*, 454(7206):877–880, August 2008.

[18] Subhash R Lele, Brian Dennis, and Frithjof Lutscher. Data cloning: easy maximum likelihood estimation for complex ecological models using Bayesian Markov chain Monte Carlo methods. *Ecology Letters*, 10(7):551–563, July 2007.

[19] C Musso, N Oudjane, and F Le Gland. Improving regularized particle filters. pages 247–271. Sequential Monte Carlo Methods in Practice, 2001.

[20] Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

[21] Radford M Neal. MCMC Using Hamiltonian Dynamics. pages 1–50. CRC Press, April 2011.

[22] Thomas B Schon. *Estimation of nonlinear dynamic systems: Theory and applications*. PhD thesis, Linköpings Universitet, 2006.

[23] Chris Snyder, Thomas Bengtsson, Peter Bickel, and Jeff Anderson. Obstacles to High-Dimensional Particle Filtering. *Monthly Weather Review*, 136(12):4629–4640, December 2008.

[24] T Toni, D Welch, N Strelkowa, A Ipsen, and M P H Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of The Royal Society Interface*, 6(31):187–202, February 2009.