Bayesian Analysis (2023) **TBA**, Number TBA, pp. 1–22

Approximate Bayesian Inference Based on Expected Evaluation

Hugo L. Hammer[*,†], Michael A. Riegler[†,§], and Håkon Tjelmeland[‡]

Abstract. Approximate Bayesian computing (ABC) and Bayesian Synthetic likelihood (BSL) are two popular families of methods to evaluate the posterior distribution when the likelihood function is not available or tractable. For existing variants of ABC and BSL, the focus is usually first put on the simulation algorithm, and after that the form of the resulting approximate posterior distribution comes as a consequence of the algorithm. In this paper we turn this around and firstly define a reasonable approximate posterior distribution by studying the distributional properties of the expected discrepancy, or more generally an expected evaluation, with respect to generated samples from the model. The resulting approximate posterior distribution will be on a simple and interpretable form compared to ABC and BSL.

Secondly a Markov chain Monte Carlo (MCMC) algorithm is developed to simulate from the resulting approximate posterior distribution. The algorithm was evaluated on a synthetic data example and on the Stepping Stone population genetics model, demonstrating that the proposed scheme has real world applicability. The algorithm demonstrates competitive results with the BSL and sequential Monte Carlo ABC algorithms, but is outperformed by the ABC MCMC.

Keywords: approximate Bayesian inference, expected evaluation, Markov chain Monte Carlo.

1 Introduction

In Bayesian inference the aim is typically the posterior distribution

$$f(\theta|x) \propto f(\theta)f(x|\theta) \tag{1}$$

where x represents observations, $f(x|\theta)$ the likelihood function and $f(\theta)$ the prior distribution representing the prior belief about θ before x was observed. Usually the likelihood function is available and tractable and most techniques to evaluate the posterior distribution relies on this.

Approximate Bayesian computation (ABC) and Bayesian synthetic likelihood (BSL) are two families of methods that generate samples from an approximation to the posterior distribution $f(\theta|x)$ even when the likelihood function $f(x|\theta)$ is not available or

[*]Department of Computer Science, Oslo Metropolitan University, Oslo, Norway, hugo.hammer@oslomet.no

[†]Department of Holistic systems, Simula Metropolitan Center of Digital Engineering, Oslo, Norway

[‡]Department of Mathematical Sciences, Norwegian University of Science and Technology, Trondheim, Norway

[§]Department of Computer Science, UiT - The Arctic University of Norway, Tromsø, Norway

https://doi.org/10.1214/23-BA1368

computationally tractable, it is only required that one is able to generate samples from $f(x|\theta)$ for any value of θ. ABC and BSL are typically applied when the likelihood represents complex simulation models and thus is intractable, such as in evolutionary biology (Csilléry et al., 2010), population genetics (Beaumont et al., 2002; Blum and François, 2010; Overcast et al., 2019; Bratsberg, 2020), econometrics (Creel, 2017), astronomy (Ishida et al., 2015) and cloud computing (Hammer et al., 2018).

The first ABC method was proposed in Pritchard et al. (1999). Marin et al. (2012) provides a nice overview of the various variants of ABC procedures that have been introduced. Since the likelihood function is intractable, ABC methods use discrepancy functions $\rho(x, \widetilde{x})$ to measure the difference between the observations x and samples $\widetilde{x}$ generated from the likelihood model $f(\widetilde{x}|\theta)$. Intuitively, if the discrepancy is small, the θ used to generate $\widetilde{x}$ should represent a reasonable sample from the posterior distribution. Usually discrepancy is measured as the distance between a statistic $s(x)$ of the observations and the same statistic of a sample $\widetilde{x}$ from $f(\widetilde{x}|\theta)$, for example

$$\rho(x, \widetilde{x}) = |s(x) - s(\widetilde{x})|, \tag{2}$$

but more advanced alternatives for the discrepancy function have also been suggested, such as using kernel functions (Blum et al., 2013), comparing the parameter estimates of an auxiliary model (Drovandi et al., 2015) or results of a classification procedure (Gutmann et al., 2018).

In the first ABC method that was introduced in Pritchard et al. (1999), possible parameter vectors are generated from the prior, $\theta \sim f(\theta)$ together with corresponding samples $\widetilde{x} \sim f(\widetilde{x}|\theta)$ from the likelihood model. If $\rho(x, \widetilde{x}) \leq \varepsilon$, the proposed θ is accepted and is a sample from an approximation to the posterior distribution,

$$\widetilde{f}_\varepsilon(\theta|x) \propto f(\theta) \int f(\widetilde{x}|\theta) I(\rho(x, \widetilde{x}) \leq \varepsilon) \mathrm{d}\widetilde{x}, \tag{3}$$

where $I(A)$ is the indicator function returning 1 if A is true and 0 otherwise. If x is a discrete variable, one may in principle use $s(x) = x$ and $\varepsilon = 0$ and the algorithm will produce exact samples from the posterior. However, the probability for acceptance is then extremely small, so in practice $\varepsilon > 0$ is used. To get a more efficient algorithm than this rejection sampling procedure, Marjoram et al. (2003) use a Metropolis–Hastings algorithm for sampling from the approximate joint posterior distribution

$$\widetilde{f}_\varepsilon(\theta, \widetilde{x}|x) \propto f(\theta) f(\widetilde{x}|\theta) I(\rho(x, \widetilde{x}) \leq \varepsilon), \tag{4}$$

from which (3) results by marginalising out $\widetilde{x}$. To avoid the need to evaluate the intractable likelihood function, potential new values for θ and $\widetilde{x}$ must be generated jointly. First, a potential new value for θ, θ', is generated from some proposal distribution given the current values of the parameter vector, and thereafter a potential new value for $\widetilde{x}$ must be generated from the likelihood model $f(\widetilde{x}|\theta')$.

Wilkinson (2013) generalises (4) by replacing the zero-one indicator function with a kernel function $K_\varepsilon(\cdot)$, where ε is the bandwidth of the kernel. One can again construct a rejection sampling algorithm to sample from the resulting approximate joint posterior distribution, but a better alternative is to use importance sampling (Blum et al., 2013). More efficient algorithms have also been developed to sample from this distribution, including Metropolis–Hastings algorithms corresponding to the one discussed above (Wegmann et al., 2009), and sequential Monte Carlo procedures (Drovandi and Pettitt, 2011; Del Moral et al., 2012).

BSL methods were introduced in Reeves and Pettitt (2005) and Gallant and McCulloch (2009). The $\widetilde{x}$ generated from $f(x|\theta)$ is in this setup used to estimate the parameter values ϕ of some parametric auxiliary model $f_A(x|\phi)$. The parameter estimate $\widehat{\phi}(\widetilde{x})$ is thereafter inserted into the auxiliary likelihood of the observations to form the resulting approximate likelihood function

$$\widetilde{f}(x|\theta) = \int f_A(x|\widehat{\phi}(\widetilde{x}))f(\widetilde{x}|\theta)d\widetilde{x} = \mathrm{E}\left[f_A(x|\widehat{\phi}(\widetilde{x})) \middle| \, x, \theta \right], \tag{5}$$

where the expectation is with respect to $\widetilde{x} \sim f(\widetilde{x}|\theta)$. The approximate posterior distribution is thereby

$$\widetilde{f}_A(\theta|x) \propto f(\theta)\mathrm{E}\left[f_A(x|\widehat{\phi}(\widetilde{x})) \middle| \, x, \theta) \right]. \tag{6}$$

The $\widehat{\phi}(\widetilde{x})$ is referred to as the binding or mapping function in the indirect inference literature (Gourieroux et al., 1993; Heggland and Frigessi, 2004). The resulting models are, in addition to Bayesian synthetic likelihood, also known as synthetic likelihood and Bayesian indirect likelihood (Wood, 2010; Drovandi et al., 2015; Price et al., 2018). To generate samples from the approximate posterior distribution various Bayesian simulation algorithms can be used. Drovandi et al. (2015) pointed out that BSL algorithms are fundamentally different from ABC in the sense that they do not rely on thresholding in any way.

Both ABC and BSL algorithms are arguably the easiest ones to understand from all sampling algorithms. When ABC was first introduced in Pritchard et al. (1999) and when Gallant and McCulloch (2009) formulated BSL, the focus was indeed on specifying sampling algorithms. The form of the resulting approximate posterior distributions in (3) and (6) were identified at a later point. For other variants of ABC and BSL the focus also lies in the simulation algorithm, and the approximate posterior distribution is then emerging as a result of the algorithm. See also the discussion in Wilkinson (2013).

In the present article we formulate an alternative strategy for defining an approximate likelihood. What we propose has naturally similarities with both ABC and BSL, but does not fall within any of these two setups. In our setup the focus is first on defining a reasonable approximate likelihood function. Only after this has been done, we begin to discuss simulation procedures for the resulting approximate posterior distribution.

The first step in our strategy is to define an evaluation function $s(x, \widetilde{x})$, which in some sense evaluates the relation between x and $\widetilde{x}$. The question is then how this evaluation function should be used to define an approximate likelihood function, since $s(x, \widetilde{x})$ is not only a function of the observations x, but also a function of the generated data $\widetilde{x}$. Indirectly $s(x, \widetilde{x})$ is also a function of θ since $\widetilde{x} \sim f(\widetilde{x}|\theta)$. Actually, there are two θ's involved in $s(x, \widetilde{x})$, the unknown true θ that we assume nature to have used when generating the observations x, and the θ that we have used to generate $\widetilde{x}$.

Likelihood functions are defined for observed data, or for functions of observed data. Except in simulation studies, it is not common to define likelihood functions for values we have generated ourselves. Instead of defining a likelihood function for $s(x, \widetilde{x})$ directly, we should therefore first use $s(x, \widetilde{x})$ to define a quantity that is only a function of x. The perhaps most natural alternative for this is the expected evaluation,

$$r(x, \theta) = \mathrm{E}[s(x, \widetilde{x})|\, x, \theta] = \int s(x, \widetilde{x}) f(\widetilde{x}|\theta) d\widetilde{x}, \tag{7}$$

where we let θ have the same unknown value as what nature used when generating the observed values in x. We should then define (an approximate) likelihood function for the expected evaluation $r(x, \theta)$.

In Section 2 we consider various evaluation functions, for all of which the expected evaluation becomes a sum of several stochastic terms. The central limit theorem can thereby be used to argue that $r(x, \theta)$ has approximately a normal distribution. There may be situations in which other limiting theorems can be used to argue for other approximate distributions. In any case, for the central limit theorem or some other limiting result to apply, the observed x must be part of an ergodic process. We note in passing that to be able to define a reasonable binding function $\widehat{\phi}(\widetilde{x})$ in the BSL setup, the same limitation applies. Combining such an approximate likelihood function with a prior for θ we get a corresponding approximate posterior distribution, that is our distribution of interest.

Just like BSL, the proposed strategy differs from ABC in that it does not rely on thresholding in any way. Like BSL, our setup includes taking an expected value with respect to the generated variable $\widetilde{x}$. However, in BSL the expectation is of an auxiliary likelihood function and the expected value is used directly as the approximate likelihood, whereas in our setup the expectation is of a more freely chosen evaluation function and the expected value is just a new stochastic variable for which an approximate likelihood function can be defined.

The remainder of this article is organised as follows. In Section 2 we detail our strategy for defining an approximate likelihood and discuss several examples of evaluation functions and corresponding expected evaluation functions. In Section 3 we propose a simulation procedure that can be used to estimate properties of the resulting approximate posterior distribution. The estimation scheme we propose requires that unbiased estimates of the approximate likelihood function is available, so in Section 4 we discuss how this can be obtained in our setup. In Section 5 we present two examples, one toy example where analytical results are available and one real data example. Finally, we provide some closing remarks in Section 6.

2 Approximate likelihood

In this section we define the setting within which we are operating in this article and formulate a novel framework for constructing an approximate likelihood to be used in an approximate Bayesian inference setting.

We consider the situation outlined in Section 1. Thus, we have observed an x, which we assume to be a realisation from some distribution $f(x|\theta)$ where θ is a vector of unknown model parameters. Adopting the Bayesian setup we assign a prior $f(\theta)$ for the model parameters, so that the posterior distribution of interest becomes as given in (1). However, we assume the likelihood function $f(x|\theta)$ to be computationally intractable so that standard techniques, like the basic Metropolis–Hastings algorithm, can not be used to sample from $f(\theta|x)$. As in ABC and BSL methods we instead assume that we are able to generate independent samples from $f(x|\theta)$ for any value of θ. For a sample $\widetilde{x} \sim f(x|\theta)$ we define an evaluation function $s(x, \widetilde{x})$, which evaluates the relation between x and $\widetilde{x}$ in some way. We discuss several possible choices of the evaluation function at the end of this section.

Given an evaluation function $s(x, \widetilde{x})$ and the corresponding expected evaluation in (7), we proceed by defining an approximate likelihood for the expected evaluation,

$$r(x, \theta)|\theta \sim g(r|\theta). \tag{8}$$

This means that for any fixed value of θ, the distribution of $r = r(x, \theta)$ is $g(r|\theta)$. We then propose to construct an approximate posterior distribution by replacing the exact likelihood $f(x|\theta)$ in (1) with $g(r(x, \theta)|\theta)$. The approximate posterior thus becomes

$$g(\theta|x) \propto f(\theta)g(r(x, \theta)|\theta). \tag{9}$$

One should note that when using $g(\theta|x)$ instead of $f(\theta|x)$ we have performed two approximations. Firstly, unless $r(x, \theta)$ is a sufficient statistic for θ in $f(x|\theta)$, it is an approximation to condition on $r(x, \theta)$ instead of x. Secondly, when the likelihood $f(x|\theta)$ is defined and an evaluation function $s(x, \widetilde{x})$ is chosen, the corresponding likelihood for $r(x, \theta)$ is also defined and can in principle be found. However, for the situations we are considering here, this likelihood function is not available, neither analytically nor numerically. The approximate likelihood $g(r|\theta)$ must therefore be chosen based on our general understanding of the situation, and for mathematical convenience.

To use the approximate Bayesian inference procedure we propose, one needs to specify an evaluation function $s(x, \widetilde{x})$ and a corresponding approximate distribution $g(r|\theta)$ for the resulting expected evaluation. As previously discussed in Section 1, the need to specify the approximate likelihood puts restrictions on what evaluation functions we can use. For example, if we select the evaluation function as $s(x, \widetilde{x}) = I(\rho(x, \widetilde{x}) \leq \varepsilon)$ then the expected evaluation becomes $r(x, \theta) = P(\rho(x, \widetilde{x}) \leq \varepsilon|x, \theta)$. Further, if we choose $g(r|\theta) \propto r$ one formally gets the same approximate posterior distribution as in the ABC setup. However, there is no reason why this $g(r|\theta)$ is a good approximation to the true distribution for $r(x, \theta)$. Choosing some simple distribution for x, $f(x|\theta)$, a discrepancy function $\rho(x, \widetilde{x})$ and a value for ε, it is of course possible to use Monte Carlo simulation to estimate the distribution for $r(x, \theta)$. We have done this Monte Carlo exercise for some cases and the distribution for $r(x, \theta)$ was then far from being proportional to $r(x, \theta)$. In general it is difficult to find an approximate distribution for $r(x, \theta)$ for such an evaluation function.

We get the same situation if we, inspired from the BSL scheme, choose $s(x, \widetilde{x}) = f_A(x|\widehat{\phi}(\widetilde{x}))$. Then the resulting expected evaluation becomes $r(x, \theta) = \widetilde{f}(x|\theta)$, where

$\widetilde{f}(x|\theta)$ is as defined in (5). By choosing $g(r|\theta) \propto r$ one formally gets the same approximate model as in the BSL setup, but there is no reason why this choice should be a good approximation to the true distribution for $r(x,\theta)$. Therefore again for this evaluation function, it is difficult to find a suitable approximate distribution for $r(x,\theta)$.

Having discussed how one should not choose the evaluation function, we will in the following present four examples of how one can define an evaluation function so that an approximate distribution for the expected evaluation is available. We start with a very simple example where the components of x are assumed to be independent and identically distributed.

Example 1. *Let $x = (x_1,\ldots,x_n)$ and $\widetilde{x} = (\widetilde{x}_1,\ldots,\widetilde{x}_n)$, and assume the elements of x and $\widetilde{x}$ to be scalar and independent and identically distributed. Let the evaluation function $s(x,\widetilde{x})$ be given as*

$$s(x,\widetilde{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i - \frac{1}{n}\sum_{i=1}^{n}\widetilde{x}_i. \tag{10}$$

The expected evaluation then becomes

$$r(x,\theta) = \frac{1}{n}\sum_{i=1}^{n} x_i - \mu, \tag{11}$$

where μ is the common mean of the x_i's and $\widetilde{x}_i$'s. Unless n is very small the central limit theorem then gives that $r(x,\theta)$ has an approximate normal distribution. As we have assumed that x and $\widetilde{x}$ both are samples from the same distribution and with the same parameter value θ, we clearly have $E[r(x,\theta)|\theta] = 0$. The variance of $r(x,\theta)$ is

$$Var[r(x,\theta)] = \frac{1}{n} Var[x_i] \tag{12}$$

and the natural estimate for $Var[x_i]$ is

$$\widehat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2, \tag{13}$$

where $\bar{x} = (1/n)\sum_{i=1}^{n} x_i$. For $g(r|\theta)$ we can thereby use

$$g(r|\theta) = \frac{1}{\sqrt{2\pi}}\frac{\sqrt{n}}{\widehat{\sigma}}\exp\left\{-\frac{nr^2}{2\widehat{\sigma}^2}\right\}. \tag{14}$$

In the next example we consider a situation where the x and $\widetilde{x}$ vectors are assumed to be parts of a stationary and ergodic process.

Example 2. *Let $x = (x_1,\ldots,x_n)$ and $\widetilde{x} = (\widetilde{x}_1,\ldots,\widetilde{x}_n)$ be part of a stationary and ergodic process so that a central limit theorem applies for the empirical covariance at lag h,*

$$c_h(x) = \frac{1}{n}\sum_{i=1}^{n-h}(x_i - \bar{x})(x_{i+h} - \bar{x}), \tag{15}$$

where $\bar{x} = (1/n)\sum_{i=1}^{n} x_i$. *Let the evaluation function be*

$$s(x, \widetilde{x}) = c_h(x) - c_h(\widetilde{x}). \tag{16}$$

Assuming $n-h$ *to be sufficient large for the empirical covariance* $c_h(\widetilde{x})$ *to be an approximately unbiased estimator for the* h *lag covariance, the expected evaluation becomes*

$$r(x, \theta) \approx c_h(x) - cov(h), \tag{17}$$

where $cov(h) = Cov[x_i, x_{i+h}]$ *is the true covariance at lag* h. *The central limit theorem then gives that* $r(x,\theta)$ *is approximately normal. As in Example 1,* x *and* $\widetilde{x}$ *are assumed to be samples from the same distribution and with the same parameter, so we clearly have* $E[r(x,\theta)|\theta] = 0$. *Moreover, we can again estimate the variance of* $r(x,\theta)$ *from the observed* x*, for example using one of the bootstrap techniques discussed in Bühlmann (2002).*

Gutmann et al. (2018) suggest to use classification in the ABC setting. In the following example we demonstrate how classification can also be implemented in our framework.

Example 3. *Let* $x = (x_1, \ldots, x_n)$ *and* $\widetilde{x} = (\widetilde{x}_1, \ldots, \widetilde{x}_n)$*, where the elements* x_i *and* $\widetilde{x}_i$ *can be scalars or vectors. We assume the elements of* x *and* $\widetilde{x}$ *to be independent and identically distributed, and let* $\eta(z; x, \widetilde{x})$ *denote the result of classifying* z *to come from the same distribution as the elements in* x *or to come from the same distribution as the elements of* $\widetilde{x}$. *The elements of* x *and* $\widetilde{x}$ *(or parts of these) are used to train the classifier, so as indicated in the notation the classification result is a function of* x *and* $\widetilde{x}$ *in addition to being a function of* z. *We assume* $\eta(z; x, \widetilde{x})$ *to return one if* z *is classified to belong to the* x *sample, and zero otherwise.*

Applying the classification procedure on each element in x *and* $\widetilde{x}$ *in turn, we let the evaluation function return the resulting number of correctly classified elements, i.e.*

$$s(x, \widetilde{x}) = \frac{1}{2n}\sum_{i=1}^{n} \eta(x_i; x, \widetilde{x}) + \frac{1}{2n}\sum_{i=1}^{n}(1 - \eta(\widetilde{x}_i; x, \widetilde{x})) \tag{18}$$

The expected evaluation then becomes

$$r(x, \theta) = \frac{1}{2n}\sum_{i=1}^{n} E[\eta(x_i; x, \widetilde{x})|x, \theta] + \frac{1}{2n}\sum_{i=1}^{n}\left(1 - E[\eta(\widetilde{x}_i; x, \widetilde{x})|x, \theta]\right). \tag{19}$$

Since the terms of $r(x,\theta)$ *are distributed on the* $[0,1]$ *interval, a natural option is to let* $g(r|\theta)$ *be a Beta distribution. However, if* n *is sufficiently large, it is again natural to let* $g(r|\theta)$ *be a normal density due to the central limit theorem. As we have assumed that* x *and* $\widetilde{x}$ *are samples from the same distribution, any reasonable classifier will perform equivalent to random guessing, so* $E[\eta(x_i; x, \widetilde{x})|\theta] = E[\eta(\widetilde{x}_i; x, \widetilde{x})|\theta] = 1/2$ *for all* i*, which gives that*

$$E[r(x, \theta)] = E[E[s(x, \widetilde{x})|x, \theta]|\theta] = E[s(x, \widetilde{x})|\theta] = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}. \tag{20}$$

In this case it is harder to estimate the variance of $r(x,\theta)$. As a first crude estimate one may use an estimate of the variance of $s(x,\widetilde{x})$ in stead, and thereafter refine the estimate of the variance as we discuss in Remark 3 below. To estimate the variance of $s(x,\widetilde{x})$ we can repeatedly split the elements of x randomly into two parts, each time use the two parts in a classification setup as above and compute the resulting evaluation value, and finally find the sample variance of these evaluation values. Letting $\widehat{\sigma}^2$ denote this sample variance it is natural to let $g(r|\theta)$ be a normal density with mean equal to a half and the variance equal to $\widehat{\sigma}^2$.

In our final example we base our evaluation function on the expression for the p-value of the two sample permutation test.

Example 4. *Let $x = (x_1, \dots, x_n)$ and $\widetilde{x} = (\widetilde{x}_1, \dots, \widetilde{x}_n)$, assume the elements of x and $\widetilde{x}$ to be independent and identically distributed, and let $T(x,\widetilde{x})$ be some test statistic that we want to use to measure the discrepancy between the elements of x and $\widetilde{x}$. Having n elements in each of x and $\widetilde{x}$, there are $N = (2n)!/((n!)^2)$ ways to redistribute the $2n$ elements in x and $\widetilde{x}$ into two new vectors of size n. We let z^k and $\widetilde{z}^k$, $k = 1, \dots, N$ denote the vectors we can construct in this way. Then let the evaluation function be equal to the p-value of the two sample permutation test,*

$$s(x,\widetilde{x}) = \frac{1}{N}\sum_{k=1}^{N} I(T(z^k,\widetilde{z}^k) \geq T(x,\widetilde{x})). \tag{21}$$

The expected evaluation then becomes

$$r(x,\theta) = \frac{1}{N}\sum_{k=1}^{N} P(T(z^k,\widetilde{z}^k) \geq T(x,\widetilde{x})|x,\theta), \tag{22}$$

where the probability is with respect to the distribution of $\widetilde{x}$, and where we should remember that z^k and $\widetilde{z}^k$ are functions of $\widetilde{x}$. Since $r(x,\theta)$ is given as an average it is again reasonable to assume $g(r|\theta)$ to be a normal density, and as we assume the elements of x and $\widetilde{x}$ to be samples from the same distribution, we can as in Example 3 use the rule of double expectation to get $E[r(x,\theta)] = 0.5$. To estimate the variance of $r(x,\theta)$ is harder also in this situation, but one may also here adopt the strategy discussed in Example 3.

We end this section with three remarks regarding the approximation framework proposed above.

Remark 1. *In the proposed framework the approximate likelihood $g(r|\theta)$ is allowed to be a function of θ. In the models where the setup are to be used, however, we think the situation is so complex that it is in practice very difficult to specify a reasonable $g(r|\theta)$ that is a function of θ. In particular, non of the $g(r|\theta)$ we consider in this article depend on θ, including the ones used in the examples discussed above. One should note that the approximate likelihood will depend on θ even if $g(r|\theta)$ is not a function of θ, since $r(x,\theta)$ is a function of θ.*

Remark 2. *One should note that in the above framework the evaluation function $s(x,\widetilde{x})$ may be a scalar or a vector. If $s(x,\widetilde{x})$ is a vector, also $r(x,\theta)$ becomes a vector of the same* *dimension and $g(r|\theta)$ becomes a multivariate distribution. To simplify the specification of a multivariate $g(r|\theta)$ one may assume the components of $r(x, \theta)$ to be independent so that $g(r|\theta)$ becomes a product of univariate distributions.*

Remark 3. *In Examples 1 and 2 good estimates of the variance of $r(x, \theta)$ are available, whereas in Examples 3 and 4 we must start with just a very crude estimate of the variance. In any case, one should note that after having explored the resulting posterior distribution one may use the simulation output to find a possibly improved estimate of the variance. To do this one should first use the simulation output to estimate a typical posterior value of the parameter vector θ. Using this value of θ one can then generate samples of x and use these to estimate the variance of $r(x, \theta)$. In Examples 1 and 2 one can simply evaluate $r(x, \theta)$ for each of the simulated x vectors and estimate the variance of $r(x, \theta)$ by the sample variance of these values. In Examples 3 and 4 one must for each generated x value also generate a set of $\widetilde{x}$ values to estimate $E[\eta(x_i; x, \widetilde{x})|x, \theta]$ and $E[\eta(\widetilde{x}_i; x, \widetilde{x})|x, \theta]$ in Example 3 and $P(T(z^k, \widetilde{z}^k) \geq T(x, \widetilde{x})|x, \theta)$ in Example 4.*

Remark 4. *A fundamental part of finding a good approximate likelihood $g(r(x, \theta)|\theta)$ is the choice of the evaluation function. In the ABC literature several methods have been suggested to find statistics that results in a good estimate of the likelihood function. Often effective (nearly sufficient) statistics can be found based on domain knowledge (Bharti et al., 2022). Other popular approaches are based on firstly finding many potentially effective statistics followed by performing dimension/statistics reduction (Blum et al., 2013). Another popular approach is to use an auxiliary model, and utilize for instance maximum likelihood estimators as statistics (Drovandi et al., 2015). These techniques can usually also be applied within our framework resulting in evaluation functions on the form in Examples 1 and 2. Effective ABC discrepancy functions suggested in the literature that are not based on statistics of the data can also in some cases be used within our framework. For instance, in Example 4 we demonstrated that the classification approach (Gutmann et al., 2018) can be used within our framework.*

3 Posterior estimation

In the previous section we proposed a framework for specifying an approximate likelihood. This framework is of course of no value unless we are able to find or estimate the properties of the resulting approximate posterior distribution. In this Section we specify an MCMC procedure that can be used to produce unbiased estimates of any property of the approximate posterior.

Assume we have the situation defined in Section 2 and that we want the posterior expectation of some function $\psi(\theta)$. We take the expectation with respect to the approximate posterior defined in (9), thus our interest is in

$$\begin{aligned}\mathcal{E}(\psi) = \mathrm{E}[\psi(\theta)|r(x, \theta)] &= \int \psi(\theta) g(\theta|x) \mathrm{d}\theta \\ &= \frac{\int \psi(\theta) f(\theta) g(r(x, \theta)|\theta) \mathrm{d}\theta}{\int f(\theta) g(r(x, \theta)|\theta) \mathrm{d}\theta}. \end{aligned} \tag{23}$$

If we had been able to sample from the approximate posterior $g(\theta|x)$ we could have estimated $\mathcal{E}(\psi)$ by a simple empirical mean of $\psi(\theta)$ for θs sampled from $g(\theta|x)$. However, as the definition of $g(\theta|x)$ includes the expected evaluation $r(x,\theta) = \mathrm{E}[s(x,\widetilde{x})|x,\theta]$ we have no simple way to generate samples from $g(\theta|x)$, not even using an MCMC procedure. In the following we adopt ideas introduced in Lyne et al. (2015) to obtain an estimate of $\mathcal{E}(\psi)$.

Assume that we for any value of θ are able to generate an unbiased estimate of $g(r(x,\theta)|\theta)$. To produce such an unbiased estimate we need to use a collection of random numbers which we denote by u. The distribution used to generate u may depend on θ and we denote the distribution by $h(u|\theta)$. Letting $\widehat{g}(r(x,\theta)|\theta,u)$ denote our unbiased estimate of the approximate likelihood we thereby have

$$g(r(x,\theta)|\theta) = \int \widehat{g}(r(x,\theta)|\theta,u)h(u|\theta)\mathrm{d}u. \tag{24}$$

A procedure for constructing such an unbiased estimate $\widehat{g}(r(x,\theta)|\theta,u)$ is discussed in Lyne et al. (2015), and in Section 4 we develop the procedure in detail when $g(r|\theta)$ is a normal distribution. One should note, however, that the $\widehat{g}(r(x,\theta)|\theta,u)$ obtained by this procedure is not necessarily non-negative, so even if it is an unbiased estimate of a density function it is not necessarily itself a density function.

Combining (23) and (24) we have

$$\mathcal{E}(\psi) = \frac{\int\int \psi(\theta)f(\theta)\widehat{g}(r(x,\theta)|\theta,u)h(u|\theta)\mathrm{d}u\,\mathrm{d}\theta}{\int\int f(\theta)\widehat{g}(r(x,\theta)|\theta,u)h(u|\theta)\mathrm{d}u\,\mathrm{d}\theta}. \tag{25}$$

To avoid the potential problem of a possible negative $\widehat{g}(r(x,\theta)|\theta,u)$ we define a new joint distribution for θ and u by

$$\breve{g}(\theta,u|x) = c(x)\cdot f(\theta)\cdot|\widehat{g}(r(x,\theta)|\theta,u)|\cdot h(u|\theta), \tag{26}$$

where

$$c(x) = \frac{1}{\int\int f(\theta)\cdot|\widehat{g}(r(x,\theta)|\theta,u)|\cdot h(u|\theta)\mathrm{d}u\mathrm{d}\theta} \tag{27}$$

is a normalising constant. Noting that we clearly have

$$\widehat{g}(r(x,\theta)|\theta,u) = |\widehat{g}(r(x,\theta)|\theta,u)|\cdot\mathrm{sign}(\widehat{g}(r(x,\theta)|\theta,u)), \tag{28}$$

we can combine (25) and (26) to get

$$\begin{aligned}
\mathcal{E}(\psi) &= \frac{\int\int \psi(\theta)f(\theta)\,|\widehat{g}(r(x,\theta)|\theta,u)|\,\mathrm{sign}\left(\widehat{g}(r(x,\theta)|\theta,u)\right)h(u|\theta)\mathrm{d}u\,\mathrm{d}\theta}{\int\int f(\theta)\,|\widehat{g}(r(x,\theta)|\theta,u)|\,\mathrm{sign}\left(\widehat{g}(r(x,\theta)|\theta,u)\right)h(u|\theta)\mathrm{d}u\,\mathrm{d}\theta} \\
&= \frac{\int\int \psi(\theta)\mathrm{sign}\left(\widehat{g}(r(x,\theta)|\theta,u)\right)\cdot c(x)f(\theta)\,|\widehat{g}(r(x,\theta)|\theta,u)|\,h(u|\theta)\mathrm{d}u\,\mathrm{d}\theta}{\int\int \mathrm{sign}\left(\widehat{g}(r(x,\theta)|\theta,u)\right)\cdot c(x)f(\theta)\,|\widehat{g}(r(x,\theta)|\theta,u)|\,h(u|\theta)\mathrm{d}u\,\mathrm{d}\theta} \\
&= \frac{\int\int \psi(\theta)\mathrm{sign}\left(\widehat{g}(r(x,\theta)|\theta,u)\right)\breve{g}(\theta,u|x)\mathrm{d}u\,\mathrm{d}\theta}{\int\int \mathrm{sign}\left(\widehat{g}(r(x,\theta)|\theta,u)\right)\breve{g}(\theta,u|x)\mathrm{d}u\,\mathrm{d}\theta}
\end{aligned}$$

$$= \frac{\mathrm{E}_{\check{g}}[\psi(\theta)\mathrm{sign}\left(\widehat{g}(r(x,\theta)|\theta,u)\right)|x]}{\mathrm{E}_{\check{g}}[\mathrm{sign}\left(\widehat{g}(r(x,\theta)|\theta,u)\right)|x]}, \tag{29}$$

where the two expectations in the last expression are with respect to $(\theta, u) \sim \check{g}(\theta, u|x)$. Thus, if we have samples $(\theta_1, u_1), \ldots, (\theta_m, u_m)$ from $\check{g}(\theta, u|x)$ a natural estimator for $\mathcal{E}(\psi)$ becomes

$$\widehat{\mathcal{E}}(\psi) = \frac{\sum_{k=1}^{m} \psi(\theta_k)\mathrm{sign}\left(\widehat{g}(r(x,\theta_k)|\theta_k,u)\right)}{\sum_{k=1}^{m} \mathrm{sign}\left(\widehat{g}(r(x,\theta_k)|\theta_k,u)\right)}. \tag{30}$$

To generate the samples from $\check{g}(\theta, u|x)$ we can use a Metropolis–Hastings algorithm. Letting θ and u denote the current values, and letting θ' and u' be corresponding potential new values, we can generate the potential new values by first generating θ' from some proposal distribution $q(\theta'|\theta)$ and thereafter generate $u' \sim h(u'|\theta')$. The acceptance probability then becomes

$$\begin{aligned}\alpha(\theta', u'|\theta, u) &= \min\left\{1, \frac{c(x)f(\theta')|\widehat{g}(r(x,\theta')|\theta',u')|h(u'|\theta')}{c(x)f(\theta)|\widehat{g}(r(x,\theta)|\theta,u)|h(u|\theta)} \cdot \frac{q(\theta|\theta')h(u|\theta)}{q(\theta'|\theta)h(u'|\theta')}\right\} \\ &= \min\left\{1, \frac{f(\theta')|\widehat{g}(r(x,\theta')|\theta',u')|q(\theta|\theta')}{f(\theta)|\widehat{g}(r(x,\theta)|\theta,u)|q(\theta'|\theta)}\right\}. \end{aligned} \tag{31}$$

4 Unbiased estimate for the approximate normal likelihood

The posterior estimation procedure described in Section 3 is based on the assumption that we are able to generate unbiased estimates for the approximate likelihood $g(r(x,\theta)|\theta)$ for any value of θ. To produce such unbiased estimates we again need to use ideas discussed in Lyne et al. (2015). In Lyne et al. (2015) the idea is described for an exponential distribution, whereas in this section we detail how it can be performed when the approximate likelihood $g(r|\theta)$ is a (scalar) normal distribution. The strategy is feasible for any (scalar) distribution for which we have available a closed form expression for the derivative of any order of the density of the distribution.

In the normal case the task is, for any value of θ, to generate independent unbiased estimates for

$$g(r(x,\theta)|\theta) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2\sigma^2}\left(r(x,\theta) - \mu\right)^2\right\}, \tag{32}$$

where μ and σ^2 have known values and $x \sim f(x|\theta)$.

We start by considering the Taylor series expansion around $t = v$ for the standard normal density,

$$\varphi(t) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{t^2}{2}\right\} = \sum_{n=0}^{\infty} \frac{\varphi^{(n)}(v)}{n!}(t-v)^n. \tag{33}$$

Introducing the Hermite polynomial of order n, which can be expressed as

$$H_n(t) = (-1)^n \frac{\varphi^{(n)}(t)}{\varphi(t)} \quad \Leftrightarrow \quad \varphi^{(n)}(t) = (-1)^n \varphi(t) H_n(t), \tag{34}$$

we get

$$\varphi(t) = \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \varphi(v) H_n(v)(t-v)^n. \tag{35}$$

Setting $t = (r(x,\theta) - \mu)/\sigma$ and $v = (r^\star - \mu)/\sigma$ for some real value $r^\star$ in the last expression, and dividing by σ we get

$$g(r(x,\theta)|\theta) = \frac{1}{\sigma}\varphi\left(\frac{r^\star - \mu}{\sigma}\right) \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} H_n\left(\frac{r^\star - \mu}{\sigma}\right)\left(\frac{r(x,\theta) - r^\star}{\sigma}\right)^n. \tag{36}$$

To obtain a fast convergence of the series in the expression above, the value of $r^\star$ should be chosen close to $r(x,\theta)$. As $r(x,\theta)$ is an unknown quantity we can not set $r^\star$ equal to $r(x,\theta)$, but we can set $r^\star$ equal to an estimate of $r(x,\theta)$. In the numerical simulations we have used

$$r^\star = \frac{1}{\nu}\sum_{i=1}^{\nu} s(x, \widetilde{x}^i), \tag{37}$$

where $\widetilde{x}^1, \ldots, \widetilde{x}^\nu$ are samples from $f(x|\theta)$, independent of each other and independent of everything else.

The sum in (36) can not be numerically evaluated since the number of terms in the sum is infinite. We can obtain an unbiased finite sum estimate of this infinite sum by truncating the infinite sum at a random stopping time τ and weighting the remaining terms appropriately. More precisely, let τ be a discrete stochastic variable with some probability mass function

$$p(\tau), \tau = 0, 1, 2, \ldots \tag{38}$$

and define an estimator for $g(r(x,\theta)|\theta)$ by

$$\widetilde{g}(r(x,\theta)|\theta) = \frac{1}{\sigma}\varphi\left(\frac{r^\star - \mu}{\sigma}\right) \sum_{n=0}^{\tau} \frac{(-1)^n}{n! P(\tau \geq n)} H_n\left(\frac{r^\star - \mu}{\sigma}\right)\left(\frac{r(x,\theta) - r^\star}{\sigma}\right)^n. \tag{39}$$

To see why this estimator is unbiased we can first note that we can write

$$\widetilde{g}(r(x,\theta)|\theta) = \frac{1}{\sigma}\varphi\left(\frac{r^\star - \mu}{\sigma}\right) \sum_{n=0}^{\infty} \frac{(-1)^n I(\tau \geq n)}{n! P(\tau \geq n)} H_n\left(\frac{r^\star - \mu}{\sigma}\right)\left(\frac{r(x,\theta) - r^\star}{\sigma}\right)^n. \tag{40}$$

We can then find the expectation of the estimator by taking the expectation for each term in the sum, and using that $\mathrm{E}[I(\tau \geq n)] = P(\tau \geq n)$ we get

$$\mathrm{E}[\widetilde{g}(r(x,\theta)|\theta)|x,\theta] = g(r(x,\theta)|\theta). \tag{41}$$

The next step in the process of constructing an unbiased estimator for $g(r(x,\theta)|\theta)$ that can be numerically computed, is to replace each of the $(r(x,\theta) - r^\star)/\sigma)$ factors in (39) by independent and unbiased estimators.

For a given value of θ we can generate independent samples $\widetilde{x} \sim f(x|\theta)$. As $r(x|\theta)$ is defined by an expectation with respect to a generated $\widetilde{x}$ we can easily generate many independent and unbiased estimators for $(r(x,\theta)-r^\star)/\sigma$. More precisely, let $\widetilde{x}^{nij}$ for $n=1,\ldots,\tau, i=1,\ldots,n$ and $j=1,\ldots,m$ for some $m\geq 1$ be independent samples from $f(x|\theta)$, and define

$$\widetilde{r}^{ni}_{\text{std}}(x,\theta)=\frac{1}{m}\sum_{j=1}^{m}\frac{s(x,\widetilde{x}^{nij})-r^\star}{\sigma} \tag{42}$$

for $n=1,\ldots,\tau$ and $i=1,\ldots,n$. The different $\widetilde{r}^{ni}_{\text{std}}(x,\theta)$ are clearly independent and from (7) we get

$$\mathrm{E}\big[\widetilde{r}^{ni}_{\text{std}}(x,\theta)\big|\, x,\theta\big]=\frac{r(x,\theta)-r^\star}{\sigma} \tag{43}$$

for each n and i. Replacing each of the $(r(x,\theta)-r^\star)/\sigma$ factors in the expression for $\widetilde{g}(r(x,\theta)|\theta)$ in (39) by one of the unbiased estimators we have now constructed, we get a new estimator for $g(r(x,\theta)|\theta)$,

$$\widehat{g}(r(x,\theta)|\theta)=\frac{1}{\sigma}\varphi\left(\frac{r^\star-\mu}{\sigma}\right)\sum_{n=0}^{\tau}\frac{(-1)^n}{n!P(\tau\geq n)}H_n\left(\frac{r^\star-\mu}{\sigma}\right)\prod_{i=1}^{n}\widehat{r}^{ni}_{\text{std}}(x,\theta). \tag{44}$$

This estimator can be computed for any value of x and θ and by construction we clearly have

$$\mathrm{E}[\widehat{g}(r(x,\theta)|\theta)|\, x,\theta]=g(r(x,\theta)|\theta). \tag{45}$$

Thus, we can use $\widehat{g}(r(x,\theta)|\theta)$ as our unbiased estimator for $g(r(x,\theta)|\theta)$ in the setup discussed in Section 3.

To complete the specification of the estimator $\widehat{g}(r(x,\theta)|\theta)$ it remains to specify the values of ν, m, and the distribution for the stopping time, $p(\tau)$. The proposed procedure works in principle for any choices for these quantities, but the choices may severely influence the computational efficiency of the procedure. To obtain computational efficiency, the estimator $\widehat{g}(r(x,\theta)|\theta)$ should have a low variance at the same time as the computational resources necessary to evaluate an estimate $\widehat{g}(r(x,\theta)|\theta)$ is low. We have experimented with different choices for ν, m and $p(\tau)$ to find quantities that give a reasonable compromise between these conflicting requirements. In the simulation experiments we used $\nu=20$. For the distribution of the stopping time τ we use

$$p(\tau)=\begin{cases}0 & \text{for } \tau=0,1,\ldots,\tau_0,\\ p\,(1-p)^{\tau-\tau_0-1} & \text{for } \tau=\tau_0+1,\tau_0+2,\ldots,\end{cases} \tag{46}$$

We used different values for m, $\tau_0\geq 0$ and $p\in(0,1)$ in the various simulation experiments, and therefore we specify what values we have used when discussing each of the examples. We end this section by noting that the variable u introduced in Section 3, which should include all random numbers used to define the estimator $\widehat{g}(r(x,\theta)|\theta)$ becomes $u=(\tau,\{\widetilde{x}^i,i=1,\ldots,\nu\},\{\widetilde{x}^{nij},n=1,\ldots,\tau,i=1,\ldots,n,j=1,\ldots,m\})$.

The algorithm based on the theory in Sections 3 and 4, is shown in Algorithm 1. By comparing the algorithm with the BSL MCMC algorithm (Price et al. (2018), Appendix B), we see that the overall structures of the algorithms are the same. Both algorithms make approximations to the likelihood by generating samples from the likelihood under the current value of the parameters θ. Both algorithms use the samples to obtain unbiased estimates of the approximate likelihood. However, the procedures to obtain the unbiased estimates are different since the approximate likelihood is different in the two models, as discussed at the end of the introduction.

Algorithm 1 Metropolis–Hastings algorithm

```
1: Input: $m, \tau_0, p, \theta_0, \nu, K, \mu, \sigma$ // See Section 2 for examples on how to find a suitable
   value for $\sigma$, e.g. using bootstrap techniques
2: Initialize:
3: $\tau \leftarrow p(\tau)$
4: $\widetilde{x}^i \leftarrow f(x|\theta_0)$ for $i = 1, \dots, \nu$
5: $r^\star \leftarrow \frac{1}{\nu} \sum_{i=1}^{\nu} s(x, \widetilde{x}^i)$
6: $\widetilde{x}^{nij} \leftarrow f(x|\theta_0)$ for $n = 1, \dots, \tau$, $i = 1, \dots, n$, $j = 1, \dots, m$
7: $u_0 \leftarrow (\tau, \{\widetilde{x}^i, i = 1, \dots, \nu\}, \{\widetilde{x}^{nij}, n = 1, \dots, \tau, i = 1, \dots, n, j = 1, \dots, m\})$
8: $\chi_0 \leftarrow \text{sign}(\widehat{g}(r(x, \theta_0)|\theta_0, u_0))$   // $\widehat{g}(r(x,\theta)|\theta, u)$ is computed using Equation (44)
9: Iterations:
10: for $k \in 1, 2, \dots, K$ do
11:     $\theta' \leftarrow q(\theta'|\theta_{k-1})$
12:     $\tau' \leftarrow p(\tau')$
13:     $\widetilde{x}'^{\,i} \leftarrow f(x|\theta_{k-1})$ for $i = 1, \dots, \nu$
14:     $r^\star \leftarrow \frac{1}{\nu} \sum_{i=1}^{\nu} s(x, \widetilde{x}'^{\,i})$
15:     $\widetilde{x}'^{\,nij} \leftarrow f(x|\theta_{k-1})$ for $n = 1, \dots, \tau$, $i = 1, \dots, n$, $j = 1, \dots, m$
16:     $u' \leftarrow (\tau', \{\widetilde{x}'^{\,i}, i = 1, \dots, \nu\}, \{\widetilde{x}'^{\,nij}, n = 1, \dots, \tau, i = 1, \dots, n, j = 1, \dots, m\})$
17:     $\alpha(\theta', u'|\theta_{k-1}, u_{k-1}) \leftarrow \min\left\{1, \frac{f(\theta')|\widehat{g}(r(x,\theta')|\theta', u')|q(\theta_{k-1}|\theta')}{f(\theta_{k-1})|\widehat{g}(r(x,\theta_{k-1})|\theta_{k-1}, u_{k-1})|q(\theta'|\theta_{k-1})}\right\}$
18:     $\nu \leftarrow U[0, 1]$   // Random number uniform on the interval $[0, 1]$
19:     if $\nu < \alpha(\theta', u'|\theta_{k-1}, u_{k-1})$ then
20:         $\theta_k \leftarrow \theta'$
21:         $u_k \leftarrow u'$
22:         $\chi_k \leftarrow \text{sign}(\widehat{g}(r(x, \theta')|\theta', u'))$
23:     else
24:         $\theta_k \leftarrow \theta_{k-1}$
25:         $u_k \leftarrow u_{k-1}$
26:         $\chi_k \leftarrow \chi_{k-1}$
27:     end if
28: end for
29: Return: $(\theta_0, u_0, \chi_0), \dots, (\theta_K, u_K, \chi_K)$
```

5 Simulation examples

In this section we first consider how the proposed procedure can be used in a toy example. Thereafter we use our scheme for the stepping stone model and compare the results with what one gets with other popular ABC and BSL algorithms.

5.1 Toy example

In the first experiment we verified convergence of the simulation algorithm to the correct posterior distribution. We let $f(x|\theta)$ in (1) be a normal distribution with unknown expectation μ and known standard deviation $\sigma = 2$. We generated $n = 250$ observations using $\mu = 0$ and used the evaluation function in Example 2 so that the likelihood function became exact. We chose a Gaussian prior with zero expectation and standard deviation equal to 5 for the unknown expectation. We ran the algorithm for a total of $5 \cdot 10^6$ iterations. The first 5 000 realisations were discarded and the remaining were used to estimate posterior expectations using (30) for different choices of ψ. The true posterior expectations were computed using numerical integration. The differences were minimal confirming that the algorithm simulated from the correct posterior distribution.

In the second experiment we evaluated computational efficiency. We let $f(x|\theta)$ be a normal distribution with unknown expectation μ and unknown standard deviation σ. We assumed μ and σ^2 to be apriori independent. The prior for μ was set to a normal distribution with zero expectation and standard deviation equal to 5. The prior distribution for σ was set to the uniform distribution on the interval $[0, 10]$.

We used the discrepancy function in Examples 1 and in Example 2 for lag $h = 0$ for mean and empirical variance, respectively. The evaluation function $s(x, \widetilde{x})$ in Example 1 will be normally distributed, while the evaluation function in Example 2 will only be approximately normal which means that our algorithm will simulate from an approximate posterior distribution for this example. The independence assumption between mean and empirical variance is however correct.

We compared the performance of our algorithm against popular and state-of-the-art algorithms in the literature, namely rejection ABC (Pritchard et al., 1999), sequential Monte Carlo (Beaumont et al., 2009; Drovandi and Pettitt, 2011; Del Moral et al., 2012), ABC MCMC (Marjoram et al., 2003; Wegmann et al., 2009), BSL and unbiased BSL (uBSL) (Price et al., 2018). All the algorithms, except BSL and uBSL, were run using the EasyABC package in R (R Core Team, 2021; Jabot et al., 2015). The BSL and uBSL algorithm, we implemented ourselves in R.[1]

Mean and empirical variance were used as summary statistics, and in the BSL and uBSL algorithms, the distribution of the two statistics was approximated with a bivariate normal distribution. For the other algorithms, a discrepancy function based on Euclidean distance $\rho(x, \widetilde{x}) = \sqrt{s_1(x, \widetilde{x})^2 + s_2(x, \widetilde{x})^2}$ was used, where $s_1(x, \widetilde{x})$ and $s_2(x, \widetilde{x})$ refer to the evaluation functions in Examples 1 and 2, respectively

Usually for such algorithms, running the generator is the computationally most demanding part of the procedure. Thus to evaluate computational efficiency, the algorithms were run until the generator was called a total of 10^5 times. We used the generated realisations to compute posterior probabilities on the grid $[1.0, 1.2, \ldots, 3.0] \times [1.0, 1.2, \ldots, 3.0]$. The true posterior probabilities were computed using numerical integration and estimation error were computed as the sum of the absolute values of the differences between the estimated and true probabilities.

[1]The algorithms are implemented in the R package BSL, but the package was not available.

Method	Error
ABC MCMC	0.09
BSL	0.10
Drovandi and Pettitt (2011)	0.10
Del Moral et al. (2012)	0.19
Suggested algorithm	**0.25**
uBSL	0.26
Beaumont et al. (2009)	0.32
Rejection ABC	0.45

Table 1: Error in absolute value estimating posterior probabilities.

For our algorithm the posterior probabilities were estimated using (30) were ψ is set to the indicator function for being inside the grid cell. We also considered a simplified estimator of (30), by setting all signs equal to 1 resulting in an ordinary empirical mean. The estimator will be biased, but with less variance, and may perform better for such short simulations.

We used $\mu = \sigma = 2$ as the true parameters and repeatedly generated datasets of size $n = 250$ to remove Monte Carlo error in the performance measures. We ran each of the algorithms for a wide range of values for the tuning parameters, to compare fairly optimal performance of the algorithms. We can imagine that pre-runs have been used to tune the parameters (Wegmann et al., 2009).

Our algorithm was run for 155 iterations (within the restriction of 10^5 calls to the generator). Table 1 shows the results. Our algorithm outperformed some of the algorithms, but Drovandi and Pettitt (2011), ABC MCMC and BSL document higher efficiency.

5.2 Stepping stone model

In this example, we evaluated the performance of our algorithm for the stepping stone population genetics model. The stepping stone model analyzes the genetic effects of migration between spatially separated subpopulations. For example, migration may limit the development of local adaption (Tufto et al., 1996).

The model studies the genetic development at L different loci for n subpopulations. In each locus we assumed two alleles, the dominant and the recessive. Let $p_{i,j,t}$ represent the portion of subpopulation i where the dominant allele at locus j is expressed at time t. This is referred to as gene frequencies below. We assumed linkage equilibrium which means that the gene frequencies in different loci changed independently of each other. The gene frequencies at locus j for each subpopulation were updated as follows in each time step

$$\breve{p}_{1,j,t+1} = (1-u)((1-m/2)p_{1,j,t} + m/2p_{2,j,t}) + uq \tag{47}$$

$$\breve{p}_{i,j,t+1} = (1-u)(m/2p_{i-1,j,t} + (1-m)p_{i,j,t} + m/2p_{i+1,j,t}) + uq, \quad i = 2,\ldots,n-1 \tag{48}$$

$$\breve{p}_{n,j,t+1} = (1-u)(m/2p_{n-1,j,t} + (1-m/2)p_{n,j,t}) + uq \tag{49}$$

where $0 < u < 1$, $0 < m < 1$ and q is a common equilibrium gene frequency for example due to migration from a large outside world. The gene frequencies were further changed by genetic drift (mutations) according to the Wright-Fisher model (Felsenstein, 2019)

$$p_{i,j,t+1} = \frac{1}{2N_i c} X_{i,j,t+1} \tag{50}$$

where N_i is the size of subpopulation i, $N_i c \in (0, N)$ the effective population size and $X_{i,j,t+1} \sim \text{Bin}(\lfloor 2N_i c \rfloor, \breve{p}_{i,j,t+1})$ where $\text{Bin}(n,p)$ is the binomial distribution with parameter n and p and $\lfloor a \rfloor$ represents the integer closest to a with a value less than or equal to a.

We considered the problem of doing inference on the model parameters u, m and c. We assume that the observations are one random outcome of the gene frequencies after the model (47) − (50) had reached equilibrium. Unfortunately, the resulting likelihood function is not analytically available or tractable and Tufto et al. (1996) suggested a multivariate normally distributed approximate likelihood function. Since it is possible to generate samples from the model, Bratsberg (2020) suggested to use ABC. We will demonstrate how the algorithm in this paper can be used to do inference on the unknown parameters, and compare the performance with other algorithms.

The equilibrium gene frequency q is a nuisance parameter. The mean gene frequency after the model reached equilibrium $\frac{1}{nL}\sum_{i=1}^{n}\sum_{j=1}^{L} p_{i,j,t}$ is approximately sufficient for q (Tufto et al., 1996). Bratsberg (2020) suggested to run the population model until the simulated mean gene frequency was equal to the observed mean gene frequency (the difference between the mean gene frequencies shifted sign). Then the generated sample should not depend on the unknown q (approximately). We followed the same procedure here.

We assumed $N_i = 250$ in each subpopulation, $n = 30$ subpopulations and $L = 30$ loci. We used uniformly distributed prior distributions on the intervals $[0.005, 1], [0, 1]$ and $[0.1, 0.5]$ for the variables u, m and c, respectively. Data was generated running the model to equilibrium with $u = 0.1$, $m = 0.5$, $c = 0.4$ and $q = 0.5$. The same values were used in Bratsberg (2020).

We used spatial autocovariance up to lag three, averaged over the different loci, as evaluation functions since they depend on the three parameters of interest, u, m and c (Bratsberg, 2020),

$$\begin{aligned} s_h(p, \widetilde{p}) = & \frac{1}{Ln}\sum_{j=1}^{L}\sum_{i=1}^{n-h}(p_{i,j} - \overline{p}_j)(p_{i+h,j} - \overline{p}_j) \\ & - \frac{1}{Ln}\sum_{j=1}^{L}\sum_{i=1}^{n-h}(\widetilde{p}_{i,j} - \overline{\widetilde{p}}_j)(\widetilde{p}_{i+h,j} - \overline{\widetilde{p}}_j), \quad h = 1,2,3 \end{aligned} \tag{51}$$

where $\overline{p}_j = \frac{1}{n}\sum_{i=1}^{n} p_{i,j}$ and p and $\widetilde{p}$ refer to the observed and generated gene frequencies, respectively. The time subscript is omitted since the evaluation function was used after the model had reached equilibrium.

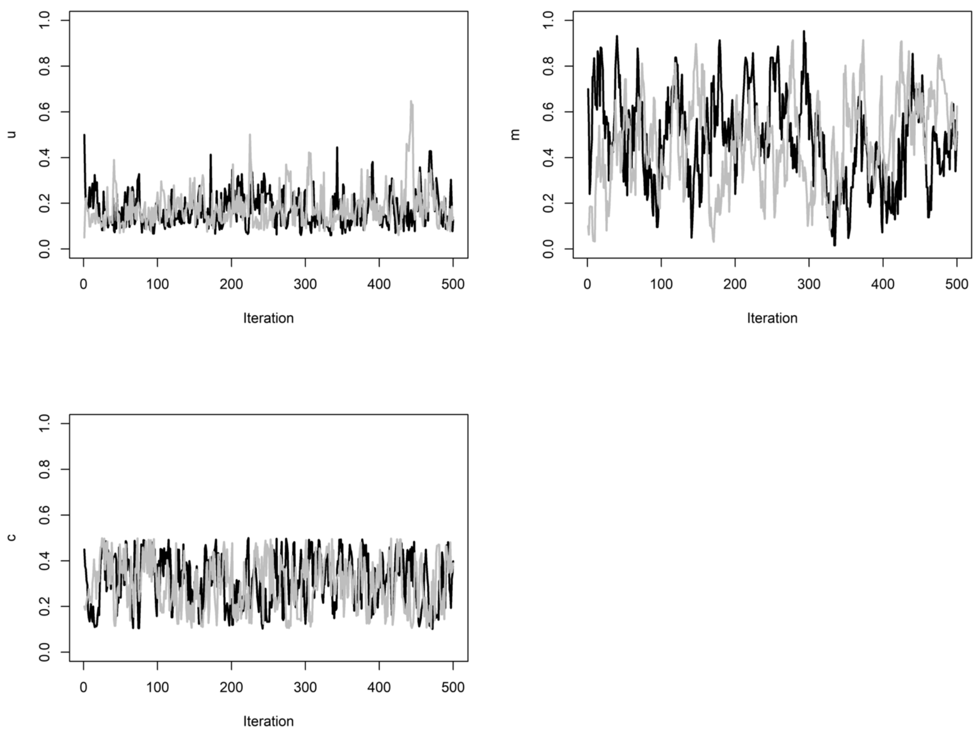

Figure 1: Trace plots of the first 500 iterations of the algorithm. The algorithm was started with small initial values, $u = 0.05, m = 0.1, c = 0.2$, and large initial values, $u = 0.5, m = 0.7, c = 0.45$, shown as gray and black curves, respectively.

In our algorithm, the expected evaluations $r(x, \theta)$ for each lag were approximated with normal distributions. We further assumed independence between the expected evaluations for different lags. The normal approximation is probably good, since the autocovariances are estimated averaging over both subpopulations and loci. However, the independence assumption is more questionable. How to estimate the variance in general is discussed in Example 2 in Section 2. However in this example, since the autocovariance function can be estimated independently for each loci, we used the independent estimates to estimate the variances.

In the suggested algorithm, we sat the standard deviations of the expected evaluations equal to 0.1 for each lag estimated as described above. For each of the unknown parameters m, u and c, we used normally distributed proposal distributions with standard deviation equal to 0.06 found by trial and error. We used $\tau_0 = 5$ and $p = 0.5$ in the stopping time distribution in (46).

Figure 1 shows trace plots of the first 500 iterations of the algorithm for each parameter. We see that the algorithm converges fast and mixes well.

Next we evaluated the consequences of the approximations of the distribution of the expected evaluation. We ran the rejection ABC algorithm with the autocovariances up

	$c \leq 0.4$		$c > 0.4$	
	$m \leq 0.5$	$m > 0.5$	$m \leq 0.5$	$m > 0.5$
$u \leq 0.1$	0.0172	0.3299	0.0422	0.0969
$u < 0.1$	0.0143	0.3906	0.0355	0.0733

Table 2: Estimated posterior probabilities using the suggested algorithm.

	$c \leq 0.4$		$c > 0.4$	
	$m \leq 0.5$	$m > 0.5$	$m \leq 0.5$	$m > 0.5$
$u \leq 0.1$	0.0125	0.3724	0.0956	0.1584
$u > 0.1$	0.0098	0.2961	0.0262	0.0290

Table 3: Estimated posterior probabilities using rejection sampling with $\varepsilon = 0.005$.

Method	Error
Del Moral et al. (2012)	0.08
ABC MCMC	0.09
Suggested algorithm	**0.16**
BSL	0.19
Drovandi and Pettitt (2011)	0.19
Rejection ABC	0.19
uBSL	0.24
Beaumont et al. (2009)	0.29

Table 4: Error in absolute value estimating the posterior probabilities conditioned on the statistic given in Table 2.

to lag three as statistics in (2) for a range of different values of the tolerance ε. Using a tolerance less than $\varepsilon = 0.005$ resulted in minimal changes in the distribution of the samples. Thus the samples from the rejection ABC algorithm with $\varepsilon = 0.005$ can in practice be seen as samples from the posterior distribution conditioning on the autocovariance statistics. The suggested algorithm in this paper will not simulate from this distribution since it uses an approximate distribution for the expected evaluation. Tables 2 and 3 show posterior probabilities for the different quadrants around the true values of the parameters. Our algorithm was run for 3100 iterations resulting in minimal Monte Carlo error in the results. Overall the posterior probabilities from the two algorithms agree fairly well. As commented above, probably the main explanation for the differences is the assumption of independence between the expected evaluation for different lags.

We further evaluated the computation efficiency of the different algorithms using the same approach as in the toy example. We measured how well the different algorithms were able to estimate the posterior probabilities conditioned on the statistics given in Table 3 using a total of 10^5 runs of the generator. The true posterior distribution is infeasible to compute for this model, and thus we compared to the posterior distribution conditioned on the statistics. The same statistics were used by all the algorithms. The results are shown in Table 4. We see that the suggested method outperforms most of the methods in the literature, but is outperformed by the sequential Monte Carlo method by Del Moral et al. (2012) and the ABC MCMC method.

6 Closing remarks

Previously developed ABC and BSL methods first focus on the simulation algorithm and the approximate posterior distribution is coming as a result of the algorithm. A consequence is that the distribution of the approximate likelihood function used in the posterior distribution involves an integral over the true (and complex) likelihood function, and the distribution of the approximate likelihood function used in the posterior distribution will be unknown. In this paper we suggest to rather start by first modeling a reasonable approximate likelihood function, and thereby a reasonable approximate posterior distribution. The resulting approximate likelihood function is both intuitive and comes with a known distribution.

We have further developed a MCMC algorithm to simulate from the approximate posterior distribution. The experiments document that the simulation algorithm has competitive computational efficiency compared BSL, uBSL and sequential Monte Carlo, but is outperformed by ABC MCMC.

References

Beaumont, M. A., Cornuet, J.-M., Marin, J.-M., and Robert, C. P. (2009). "Adaptive approximate Bayesian computation." *Biometrika*, 96(4): 983–990. MR2767283. doi: https://doi.org/10.1093/biomet/asp052. 15, 16, 19

Beaumont, M. A., Zhang, W., and Balding, D. J. (2002). "Approximate Bayesian computation in population genetics." *Genetics*, 162(4): 2025–2035. 2

Bharti, A., Filstroff, L., and Kaski, S. (2022). "Approximate Bayesian Computation with Domain Expert in the Loop." In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 1893–1905. PMLR.
URL https://proceedings.mlr.press/v162/bharti22a.html 9

Blum, M. G. and François, O. (2010). "Non-linear regression models for approximate Bayesian computation." *Statistics and Computing*, 20(1): 63–73. MR2578077. doi: https://doi.org/10.1007/s11222-009-9116-0. 2

Blum, M. G., Nunes, M. A., Prangle, D., Sisson, S. A., et al. (2013). "A comparative review of dimension reduction methods in approximate Bayesian computation." *Statistical Science*, 28(2): 189–208. MR3112405. doi: https://doi.org/10.1214/12-sts406. 2, 3, 9

Bratsberg, A. M. (2020). "Inference of migration patterns from genetic data using approximate Bayesian computation." Master's thesis, Department of Mathematical Sciences, Norwegian University of Science and Technology. 2, 17

Bühlmann, P. (2002). "Bootstraps for time series." *Statistical science*, 52–72. MR1910074. doi: https://doi.org/10.1214/ss/1023798998. 7

Creel, M. (2017). "Neural nets for indirect inference." *Econometrics and Statistics*, 2: 36–49. 2

Csilléry, K., Blum, M. G., Gaggiotti, O. E., and François, O. (2010). "Approximate Bayesian computation (ABC) in practice." *Trends in ecology & evolution*, 25(7): 410–418. 2

Del Moral, P., Doucet, A., and Jasra, A. (2012). "An adaptive sequential Monte Carlo method for approximate Bayesian computation." *Statistics and Computing*, 22(5): 1009–1020. MR2950081. doi: https://doi.org/10.1007/s11222-011-9271-y. 3, 15, 16, 19

Drovandi, C. C. and Pettitt, A. N. (2011). "Estimation of parameters for macroparasite population evolution using approximate Bayesian computation." *Biometrics*, 67(1): 225–233. MR2898834. doi: https://doi.org/10.1111/j.1541-0420.2010.01410.x. 3, 15, 16, 19

Drovandi, C. C., Pettitt, A. N., Lee, A., et al. (2015). "Bayesian indirect inference using a parametric auxiliary model." *Statistical Science*, 30(1): 72–95. MR3317755. doi: https://doi.org/10.1214/14-STS498. 2, 3, 9

Felsenstein, J. (2019). "Theoretical evolutionary genetics." *University of Washington, Seattle*. 17

Gallant, A. R. and McCulloch, R. E. (2009). "On the determination of general scientific models with application to asset pricing." *Journal of the American Statistical Association*, 104(485): 117–131. MR2663037. doi: https://doi.org/10.1198/jasa.2009.0008. 3

Gourieroux, C., Monfort, A., and Renault, E. (1993). "Indirect inference." *Journal of applied econometrics*, 8(S1): S85–S118. 3

Gutmann, M. U., Dutta, R., Kaski, S., and Corander, J. (2018). "Likelihood-free inference via classification." *Statistics and Computing*, 28(2): 411–425. MR3747571. doi: https://doi.org/10.1007/s11222-017-9738-6. 2, 7, 9

Hammer, H. L., Yazidi, A., Bratterud, A., Haugerud, H., and Feng, B. (2018). "A queue model for reliable forecasting of future CPU consumption." *Mobile Networks and Applications*, 23(4): 840–853. 2

Heggland, K. and Frigessi, A. (2004). "Estimating functions in indirect inference." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(2): 447–462. MR2062387. doi: https://doi.org/10.1111/j.1369-7412.2003.05341.x. 3

Ishida, E., Vitenti, S., Penna-Lima, M., Cisewski, J., de Souza, R., Trindade, A., Cameron, E., Busti, V., collaboration, C., et al. (2015). "Cosmoabc: likelihood-free inference via population Monte Carlo approximate Bayesian computation." *Astronomy and Computing*, 13: 1–11. 2

Jabot, F., Faure, T., Dumoulin, N., and Albert., C. (2015). *EasyABC: Efficient approximate Bayesian computation sampling schemes*. R package version 1.5.
URL https://CRAN.R-project.org/package=EasyABC 15

Lyne, A.-M., Girolami, M., Atchadé, Y., Strathmann, H., Simpson, D., et al. (2015). "On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods." *Statistical science*, 30(4): 443–467. MR3432836. doi: `https://doi.org/10.1214/15-STS523`. 10, 11

Marin, J.-M., Pudlo, P., Robert, C. P., and Ryder, R. J. (2012). "Approximate Bayesian computational methods." *Statistics and Computing*, 22(6): 1167–1180. MR2992292. doi: `https://doi.org/10.1007/s11222-011-9288-2`. 2

Marjoram, P., Molitor, J., Plagnol, V., and Tavaré, S. (2003). "Markov chain Monte Carlo without likelihoods." *Proceedings of the National Academy of Sciences*, 100(26): 15324–15328. 2, 15

Overcast, I., Emerson, B. C., and Hickerson, M. J. (2019). "An integrated model of population genetics and community ecology." *Journal of Biogeography*, 46(4): 816–829. 2

Price, L. F., Drovandi, C. C., Lee, A., and Nott, D. J. (2018). "Bayesian synthetic likelihood." *Journal of Computational and Graphical Statistics*, 27(1): 1–11. MR3788296. doi: `https://doi.org/10.1080/10618600.2017.1302882`. 3, 13, 15

Pritchard, J. K., Seielstad, M. T., Perez-Lezaun, A., and Feldman, M. W. (1999). "Population growth of human Y chromosomes: a study of Y chromosome microsatellites." *Molecular biology and evolution*, 16(12): 1791–1798. 2, 3, 15

R Core Team (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria.
URL `https://www.R-project.org/` 15

Reeves, R. and Pettitt, A. (2005). "A theoretical framework for approximate Bayesian computation." In *Proceedings of the 20th International Workshop for Statistical Modelling, Sydney Australia*, 393–396. 3

Tufto, J., Engen, S., and Hindar, K. (1996). "Inferring patterns of migration from gene frequencies under equilibrium conditions." *Genetics*, 144(4): 1911–1921. 16, 17

Wegmann, D., Leuenberger, C., and Excoffier, L. (2009). "Efficient approximate Bayesian computation coupled with Markov chain Monte Carlo without likelihood." *Genetics*, 182(4): 1207–1218. 3, 15, 16

Wilkinson, R. D. (2013). "Approximate Bayesian computation (ABC) gives exact results under the assumption of model error." *Statistical applications in genetics and molecular biology*, 12(2): 129–141. MR3071024. doi: `https://doi.org/10.1515/sagmb-2013-0010`. 2, 3

Wood, S. N. (2010). "Statistical inference for noisy nonlinear ecological dynamic systems." *Nature*, 466(7310): 1102–1104. 3