



**HAL**  
open science

## Adaptive approximate Bayesian computation for complex models

Maxime Lenormand, Franck Jabot, Guillaume Deffuant

► **To cite this version:**

Maxime Lenormand, Franck Jabot, Guillaume Deffuant. Adaptive approximate Bayesian computation for complex models. *Computational Statistics*, 2013, 28 (6), pp.2777-2796. 10.1007/s00180-013-0428-3 . hal-00638484v4

**HAL Id: hal-00638484**

**<https://hal.science/hal-00638484v4>**

Submitted on 20 May 2015

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Adaptive approximate Bayesian computation for complex models

Maxime Lenormand,<sup>1</sup> Franck Jabot,<sup>1</sup> and Guillaume Deffuant<sup>1</sup>

<sup>1</sup>*IRSTEA, LISC, 24 avenue des Landais, 63172 AUBIERE, France*

We propose a new approximate Bayesian computation (ABC) algorithm that aims at minimizing the number of model runs for reaching a given quality of the posterior approximation. This algorithm automatically determines its sequence of tolerance levels and makes use of an easily interpretable stopping criterion. Moreover, it avoids the problem of particle duplication found when using a MCMC kernel. When applied to a toy example and to a complex social model, our algorithm is 2 to 8 times faster than the three main sequential ABC algorithms currently available.

Approximate Bayesian computation (ABC) techniques appear particularly relevant for calibrating stochastic models because they are easy to implement and applicable to any model. They generate a sample of model parameter values  $(\theta_i)_{i=1,\dots,N}$  (often also called particles) from the prior distribution  $\pi(\theta)$  and select the  $\theta_i$  values leading to model outputs  $x \sim f(x|\theta_i)$  satisfying a proximity criterion with the target data  $y$  ( $\rho(x, y) \leq \epsilon$ ,  $\rho$  expressing a distance,  $\epsilon$  being a tolerance level). The selected sample of parameter values approximates the posterior distribution of parameters, leading to model outputs with the expected quality of approximation. However, in practice, running these techniques is very demanding computationally because sampling the whole space of parameters requires a number of simulations which grows exponentially with the number of parameters to identify. This tends to limit the application of these techniques to easily computable models [1]. In this paper, our goal is minimizing the number of model runs for reaching a given quality of posterior approximation, and thus to make the approach applicable to a larger set of models.

ABC is the subject of intense scientific researches and several improved versions of the original scheme are available, such as using local regressions to improve parameter inference [2, 3], automatically selecting informative summary statistics [4, 5], coupling to Markov chain Monte Carlo [6, 7] or improving sequentially the posterior distributions with sequential Monte Carlo methods [8–10]. This last class of methods approximates progressively the posterior, using sequential samples  $S^{(t)} = (\theta_i^{(t)})_{i=1,\dots,N}$  derived from sample  $S^{(t-1)}$ , and using a decreasing set of tolerance levels  $\{\epsilon_1, \dots, \epsilon_T\}$ . This strategy focuses the sampling effort in parts of the parameter space of high likelihood, avoiding to spend much computing time in systematically sampling the whole parameter space.

The first sequential method applied to ABC was proposed by [8] with the ABC-PRC (Partial Rejection Control). This method is based on a theoretical work of [11] to ABC. However, in [10] the authors have shown that this method leads to a bias in the approximation of the posterior. In [9, 10] the authors proposed a new algorithm, called Population Monte Carlo ABC in [10] and hereafter called PMC. This algorithm, corrects the bias by assigning to each particle a weight corresponding to the inverse of its importance in the sample. It is particularly interesting

in our perspective because it provides with a rigorous framework to the sequential sample idea, which seems a good way for minimizing the number of runs. In this approach, the problem is then defining the sequence of tolerance levels  $\{\epsilon_1, \dots, \epsilon_T\}$ . In [12] and [13] the authors solve partly this problem by deriving the tolerance level at a given step from the previously selected sample. However, a difficulty remains: when to stop? If the final tolerance level  $\epsilon_T$  is too large, the final posterior will be of bad quality. Inversely, a too small  $\epsilon_T$  leads to a posterior that could have been obtained with less model runs.

In this paper, we propose a modification of the population Monte Carlo ABC algorithm proposed in [10] that we call adaptive population Monte Carlo ABC (hereafter called APMC). This new algorithm determines by itself the sequence of tolerance levels as in [12] and [13], and it also provides a stopping criterion. Furthermore, our approach avoids the problem of duplications of particles due to the MCMC kernel used in [12] and [13]. We prove that the computation of the weights associated to the particles in this algorithm lead to the intended posterior distribution and we also prove that the algorithm stops whatever the chosen value of the stopping parameter. We show that our algorithm, applied to a toy example and to an individual-based social model, requires significantly less simulations to reach a given quality level of the posterior distribution than the population Monte Carlo ABC algorithm of [10] (hereafter called PMC), the replenishment SMC ABC algorithm of [12] (hereafter called RSMC) and the adaptive SMC ABC algorithm of [13] (hereafter called SMC). Our new algorithm has been implemented in the R package 'EasyABC' [14].

## Sequential Monte-Carlo methods in approximate Bayesian computation

In this section we present the three main sequential ABC algorithms currently available and their limitations. We present the Population Monte-Carlo ABC proposed in [10] (hereafter called PMC), the Replenishment Sequential Monte-Carlo ABC proposed in [12] and the Sequential Monte-Carlo ABC proposed in [13]. These algorithms are detailed in Appendix A.

### The PMC algorithm

This method consists in generating a sample  $S^{(t)} = (\theta_i^{(t)})_{i=1,\dots,N}$  at each iteration of the algorithm,  $1 \leq t \leq T$ . Each particle of the sample  $S^{(t)}$  satisfying the predefined tolerance level  $\epsilon_t$  where  $\epsilon_1 \geq \epsilon_t \geq \epsilon_T$ . We say that a parameter value  $\theta_i^{(t)}$ , satisfies the tolerance level  $\epsilon_t$ , if when running the model we get  $x \sim f(x|\theta_i^{(t)})$ , such that its distance  $\rho_i^{(t)} = \rho(x, y)$  to the target data  $y$ , is below  $\epsilon_t$ . At step  $t$  the sample  $S^{(t)}$  is derived from sample  $S^{(t-1)}$  using a particle filter methodology. The first sample  $S^1$  is generated using a regular ABC step. At step  $t$  a new particle  $\theta_i^{(t)}$  is generated using a Markov transition kernel  $K_t$ ,  $\theta_i^{(t)} \sim K_t(\theta|\theta^*)$ , until  $\theta_i^{(t)}$  satisfies  $\epsilon_t$  where  $\theta^*$  is randomly draw from  $S^{(t-1)}$  with probability  $(w_i^{(t-1)})_{i=1,\dots,N}$ . The weight  $w_i^{(t-1)}$  is proportional to the inverse of its importance in the sample  $S^{(t-1)}$  (Eq. 2). The kernel function  $K_t$  is a Gaussian kernel with a variance equal to twice the weighted empirical variance of the set  $S^{(t-1)}$  [10]. The algorithm stops when the sample  $S^{(T)}$  is generated i.e the target  $\epsilon_T$  is reached. See Algorithm 2 for details.

#### Weights correcting the kernel sampling bias

As pointed out by [10], the newly generated particles  $\theta_i^{(t)}$  in a sequential procedure are no more drawn from the prior distribution but from a specific probability density  $d_i^{(t)}$  that depends on the particles selected at the previous step and on the chosen kernel. This introduces a bias in the procedure. This bias should be corrected by attributing a weight equal to  $\pi(\theta_i^{(t)})/d_i^{(t)}$  to each newly generated particle  $\theta_i^{(t)}$ .

The density of probability  $d_i^{(t)}$  to generate particle  $\theta_i^{(t)}$  at step  $t$  is given by the sum of the probabilities to reach  $\theta_i^{(t)}$  from one of the  $N$  particles of the previous step times their respective weights:

$$d_i^{(t)} \propto \sum_{j=1}^N w_j^{(t-1)} \sigma_{t-1}^{-1} \varphi \left( \sigma_{t-1}^{-1} (\theta_i^{(t)} - \theta_j^{(t-1)}) \right) \quad (1)$$

where  $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$  is the kernel function.

This yields the expression of the weight  $w_i^{(t)}$  to be attributed to the newly drawn particle  $\theta_i^{(t)}$ :

$$w_i^{(t)} \propto \frac{\pi(\theta_i^{(t)})}{\sum_{j=1}^N w_j^{(t-1)} \sigma_{t-1}^{-1} \varphi \left( \sigma_{t-1}^{-1} (\theta_i^{(t)} - \theta_j^{(t-1)}) \right)} \quad (2)$$

#### Limitations of the PMC algorithm

The major problem in the PMC algorithm is to define the decreasing sequence of tolerance levels

$\{\epsilon_1, \dots, \epsilon_T\}$  to get close to an optimal gain in computing time. If the decrease in tolerance values is too sharp or too shallow, the benefits of the importance sampling procedure has good chance to be lower than what could be possible. In the following, we will indeed demonstrate that our algorithm leads to a sequence of tolerance levels which clearly outperforms an arbitrary choice for the sequence of tolerance levels.

### The RSMC and the SMC algorithms

In [12] and [13] the authors proposed two methods to determine "on-line" the sequence of tolerance levels. The main idea is to define the  $\epsilon_t$  value with the previous sample  $S^{(t-1)}$ . In the RSMC algorithm of [12],  $\epsilon_t$  is defined as a quantile of the  $\rho(x, y)$  values of the previous sample  $S^{(t-1)}$  (see Algorithm 3 for details). In the SMC algorithm of [13],  $\epsilon_t$  is computed so that the effective sample size of the particles is reduced by a constant factor at each time step (see Algorithm 4 for details).

A second difference between the PMC and the RSMC/SMC algorithms concerns the proposal distribution. The RSMC and the SMC algorithms use a MCMC kernel to move the particles. At step  $t$ , a new particle  $\theta_i^{(t)}$  is generated using a MCMC kernel  $\theta_i^{(t)} \sim K_t(\theta|\theta^*)$  where  $\theta^*$  is randomly draw from  $S^{(t-1)}$  with probability  $(w_i^{(t-1)})_{i=1,\dots,N}$ . This weight  $(w_i^{(t)})_{i=1,\dots,N}$  is equal to 1 if the particle  $\theta_i^{(t)}$  satisfies  $\epsilon_t$ , and to 0 otherwise. The jump is accepted with probability,  $p_{acc}$ , based on the Metropolis-Hastings ratio (Eq. 3).

$$1 \wedge \frac{\pi(\theta_i^{(t)}) K_t(\theta^*|\theta_i^{(t)})}{\pi(\theta^*) K_t(\theta_i^{(t)}|\theta^*)} \mathbb{1}_{\rho(x,y) \leq \epsilon_t} \quad (3)$$

where  $x \wedge y$  means the minimum of  $x$  and  $y$ .

#### Limitations of the RSMC and the SMC algorithms

The MCMC kernel used in [12] and [13] to sample new values  $\theta_j^{(t)}$  has a significant drawback in our view: it can lead to particle duplications. Indeed, each time the MCMC jumps from a particle to a new one which is not accepted, the initial particle is kept in the new sample of particles. When this occurs several times with the same initial particle, this particle appears several times in the new sample. The number of such "duplicated" particles can grow and strongly deteriorate the quality of the posterior, as illustrated below. To solve this problem, [12] proposed to perform  $R$  MCMC jump trials instead of one.  $R$  evolves during the course of the algorithm (Eq. 4) since its value is chosen such that there is a probability of  $1 - c$  that the particle gets moved at least once where  $c = 0.01$  in

[12]. To circumvent the problem of particle duplications [13] proposed to resample the parameter values when too many are duplicated. In [13] the authors also proposed to run the model  $M$  times for each particle, in order to decrease the variance of the acceptance ratio of the MCMC jump. However, all these solutions increase the number of model runs, going against the initial benefit of using sequential samples.

$$R = \frac{\log(c)}{\log(1 - p_{acc})} \quad (4)$$

### Adaptive population Monte-Carlo approximate Bayesian computation

#### Overview of the APMC algorithm

The APMC algorithm follows the main principles of the sequential ABC, and defines on-line the tolerance level at each step like in [15], [12] and [13]. For each tolerance level  $\epsilon_t$ , it generates a sample  $S^{(t)}$  of particles and computes their associated weights. This weighted sample approximates the posterior distribution, with an increasing approximation quality as  $\epsilon_t$  decreases. Suppose the APMC reached step  $t - 1$ , with a sample  $S^{(t-1)}$  of  $N_\alpha = \lfloor \alpha N \rfloor$  particles and their associated weights  $(\theta_i^{(t-1)}, w_i^{(t-1)})_{i=1, \dots, N_\alpha}$ , the main features of the APMC are (see Algorithm 5 for details):

- the algorithm generates  $N - N_\alpha$  particles  $(\theta_j^{(t-1)})_{j=N_\alpha+1, \dots, N}$  where  $\theta_j^{(t-1)} \sim \mathcal{N}(\theta_j^*, \sigma_{(t-1)}^2)$ , the seed  $\theta_j^*$  is randomly drawn from the weighted set  $(\theta_i^{(t-1)}, w_i^{(t-1)})_{i=1, \dots, N_\alpha}$  and the variance  $\sigma_{(t-1)}^2$  of the Gaussian kernel  $\mathcal{N}(\theta_j^*, \sigma_{(t-1)}^2)$  is twice the empirical variance of the weighted set  $(\theta_i^{(t-1)}, w_i^{(t-1)})_{i=1, \dots, N_\alpha}$ , following [10].
- the weights  $w_j^{(t-1)}$  of the new particles  $(\theta_j^{(t-1)})_{j=N_\alpha+1, \dots, N}$  are computed so that these new particles can be combined with the sample  $S^{(t-1)}$  of the previous step without causing a bias in the posterior distribution. These weights are given by Eq. 6 (see below).
- the algorithm concatenates the  $N_\alpha$  previous particles  $(\theta_i^{(t-1)})_{i=1, \dots, N_\alpha}$  with the  $N - N_\alpha$  new particles  $(\theta_j^{(t-1)})_{j=N_\alpha+1, \dots, N}$ , together with their associated weights and distances to the data. This constitutes a new set noted  $S_{temp}^{(t)} = (\theta_i^{(t)}, w_i^{(t)}, \rho_i^{(t)})_{i=1, \dots, N}$ .
- the next tolerance level  $\epsilon_t$  is determined as the first  $\alpha$ -quantile of the  $(\rho_i^{(t)})_{i=1, \dots, N}$ .

- the new sample  $S^{(t)} = (\theta_i^{(t)}, w_i^{(t)})_{i=1, \dots, N_\alpha}$  is then constituted from the  $N_\alpha$  particles of  $S_{temp}^{(t)}$  satisfying the tolerance level  $\epsilon_t$ .
- if the proportion  $p_{acc}$  of particles satisfying the tolerance level  $\epsilon_{t-1}$  among the  $N - N_\alpha$  newly generated particles is below a chosen value  $p_{acc_{min}}$ , the algorithm stops, and its result is  $(\theta_i^{(t)})_{i=1, \dots, N_\alpha}$  with their associated weights.

Note that in our algorithm, to get a number  $N_\alpha$  of retained particles for the next step, the choice of  $\epsilon_t$  is heavily constrained: it has to be at least equal to the first  $\alpha$ -quantile of the  $(\rho_i^{(t)})_{i=1, \dots, N}$  and smaller than the immediately superior  $(\rho_i^{(t)})$  value. We chose to fix it to the first  $\alpha$ -quantile for simplicity. This choice also ensures that the tolerance level decreases from one iteration to the next: in the worst case where  $p_{acc} = 0$  (no newly simulated particles accepted),  $\epsilon_t = \epsilon_{t-1}$ . Our algorithm does not use a MCMC kernel and avoids duplicating particles. It requires a reweighting step in  $O(N_\alpha^2)$  instead of  $O(N_\alpha)$  in [12], but in our perspective, this computational cost is supposed negligible compared with the cost of running the model.

#### Weights correcting the kernel sampling bias

For the APMC algorithm the density of probability  $d_i^{(t)}$  to generate particle  $\theta_i^{(t)}$  at step  $t$  is:

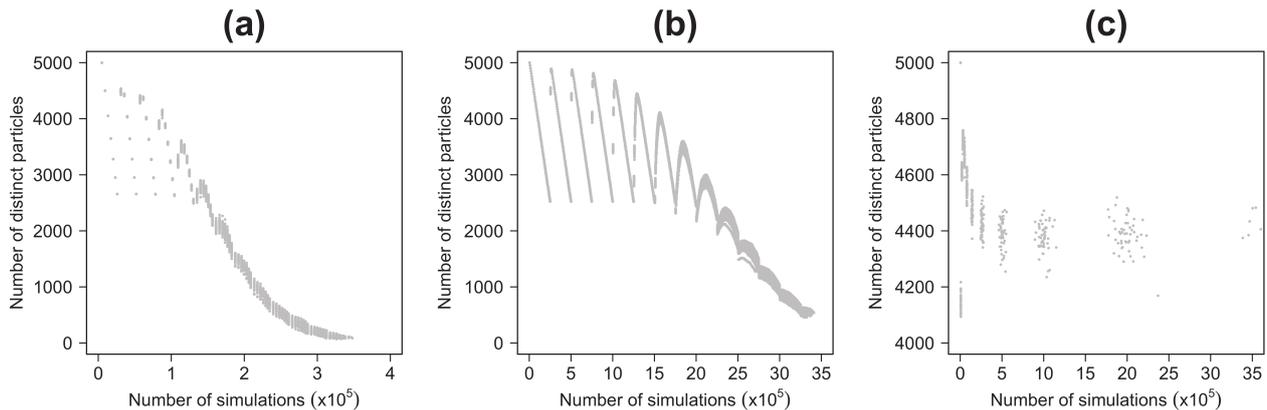
$$d_i^{(t)} = \sum_{j=1}^{N_\alpha} \frac{w_j^{(t-1)}}{\sum_{k=1}^{N_\alpha} w_k^{(t-1)}} \sigma_{t-1}^{-1} \varphi \left( \sigma_{t-1}^{-1} (\theta_i^{(t)} - \theta_j^{(t-1)}) \right) \quad (5)$$

where  $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$  is the kernel function.

This yields the expression of the weight  $w_i^{(t)}$  to be attributed to the newly drawn particle  $\theta_i^{(t)}$ :

$$w_i^{(t)} = \frac{\pi(\theta_i^{(t)})}{\sum_{j=1}^{N_\alpha} \left( w_j^{(t-1)} / \sum_{k=1}^{N_\alpha} w_k^{(t-1)} \right) \sigma_{t-1}^{-1} \varphi \left( \sigma_{t-1}^{-1} (\theta_i^{(t)} - \theta_j^{(t-1)}) \right)} \quad (6)$$

This formula differs from the scheme of [10] where the weights need only to be proportional to Eq. 6 at each step. Since we want to concatenate particles obtained at different steps of the algorithm (while [10] generate the sample at step  $t$  from scratch), we need the scaling of weights to be consistent across the different steps of the algorithm. Using the weight of Eq. 6 guarantees the correction of the sampling bias throughout the APMC procedure and ensures that the  $N_\alpha$  weighted particles  $\theta_i^{(t)}$  produced at the  $t$ -th iteration follow the posterior distribution  $\pi(\theta | \rho(x, y) < \epsilon_t)$ .



**Figure 1:** Number of distinct particles in a sample of  $N = 5000$  particles during the course of the SMC and RSMC algorithms applied to the toy example; In all three panels we plot the results obtained for 50 executions of the algorithm. (a) SMC with  $\alpha = 0.9$  and  $M = 1$ ; (b) SMC with  $\alpha = 0.99$  and  $M = 1$ ; (c) RSMC with  $\alpha = 0.5$ . In all three panels, the tolerance target is equal to 0.001.

### The stopping criterion

We stop the algorithm when the proportion of "accepted" particles (Eq. 7) among the  $N - N_\alpha$  new particles is below a predetermined threshold  $p_{acc_{min}}$ . This choice of stopping rule ensures that additional simulations would only marginally change the posterior distribution. Note that this stopping criterion will be achieved even if  $p_{acc_{min}} = 0$ , this ensures that the algorithm converges. We present a formal proof of this assertion in Appendix B.

$$p_{acc}(t) = \frac{1}{N - N_\alpha} \sum_{k=N_\alpha+1}^N \mathbb{1}_{\rho_k^{(t-1)} < \epsilon_{t-1}} \quad (7)$$

### Experiments on a toy example

We consider four algorithms: APMC, PMC, the SMC and the RSMC. Their implementations in R [16] are available [21]. We compare them on the toy example studied in [8] where  $\pi(\theta) = \mathcal{U}_{[-10,10]}$  and  $f(x|\theta) \sim \frac{1}{2}\phi(\theta, \frac{1}{100}) + \frac{1}{2}\phi(\theta, 1)$  where  $\phi(\mu, \sigma^2)$  is the normal density of mean  $\mu$  and variance  $\sigma^2$ . In this example, we consider that  $y = 0$  is observed, so that the posterior density of interest is proportional to  $(\phi(0, \frac{1}{100}) + \phi(0, 1))\pi(\theta)$ .

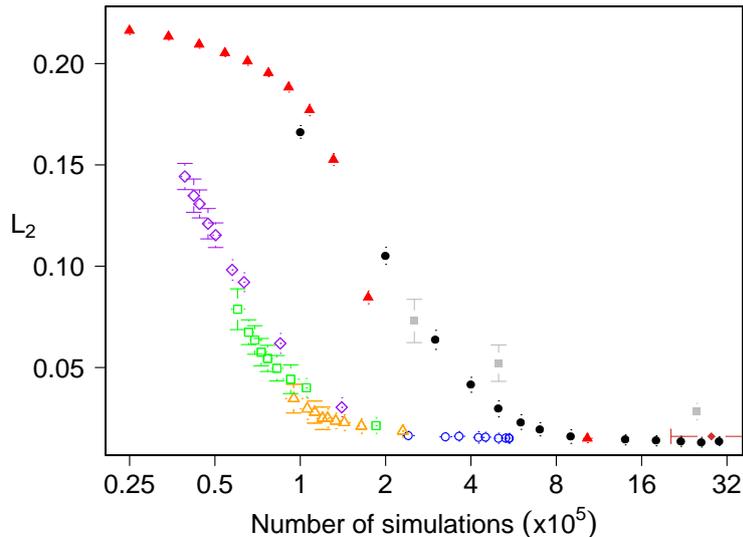
We structure the comparisons on two indicators: the number of simulations performed during the application of the algorithms, and the  $\mathbb{L}_2$  distance between the exact posterior density and the histogram of particle values obtained with the algorithms. This  $\mathbb{L}_2$  distance is computed on the 300-tuple obtained by dividing the support  $[-10, 10]$  into 300 equally-sized bins. We choose the  $\mathbb{L}_2$  distance to compare the sample to the true posterior because it is a well-known accuracy measure easy to compute and a good indicator to compare different methods.

We choose  $N = 5000$  particles and a target tolerance level equal to 0.01. For the PMC algorithm we use a decreasing sequence of tolerance levels from  $\epsilon_1 = 2$  down to  $\epsilon_{11} = 0.01$ . For the SMC algorithm, we use 3 different values for  $\alpha$ :  $\{0.9, 0.95, 0.99\}$  and  $M = 1$  as in [13]. For the RSMC algorithm we use  $\alpha = 0.5$  as in [12]. To explore our algorithm, we test 9 different values for  $\alpha$ :  $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ , and 4 different values for  $p_{acc_{min}}$ :  $\{0.01, 0.05, 0.1, 0.2\}$ . In each case, we perform 50 times the algorithm, and compute the average and standard deviation of the two indicators: the total number of simulations and the  $\mathbb{L}_2$  distance between the exact posterior density and the histogram of particle values. We used as kernel transition a normal distribution parameterized with twice the weighted variance of the previous sample, as in [10].

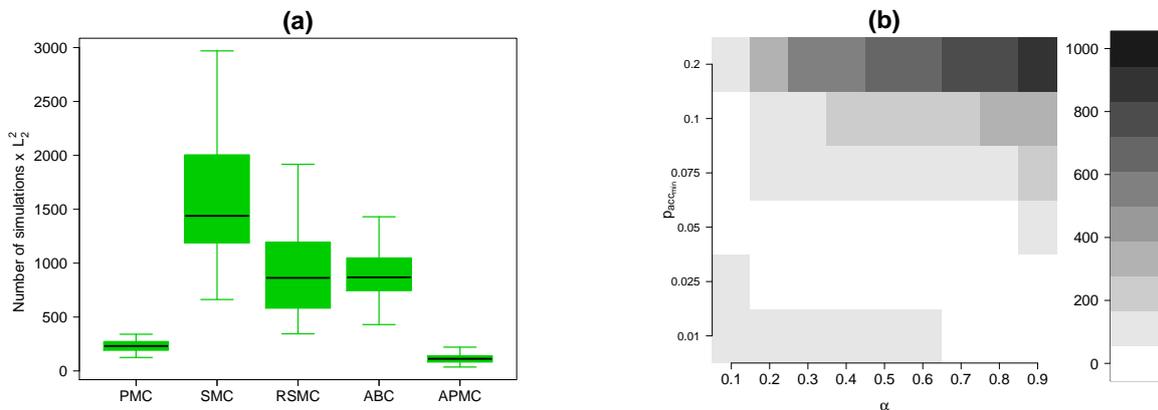
We report below the effects of varying  $\alpha$  and  $p_{acc_{min}}$  on the performance of our algorithm, and compare it with the PMC, SMC and RSMC algorithms.

### Particle duplication in SMC and RSMC

The number of distinct particles decreases during the course of the SMC algorithm whatever the value of  $\alpha$ , as shown on Fig. 1a-b. The oscillations of the number of distinct particles are caused by the resampling step in the SMC algorithm (see [13]), but they are not sufficient to counterbalance the overall decrease. This decrease deteriorates the posterior approximation as shown on Fig. 2. For the RSMC algorithm, the initial oscillation of the number of particles is due to the initial value of  $R$ , initially set to 1, but which quickly evolves towards a value ensuring a relatively constant number of distinct particles. This number of distinct particles is maintained at a reasonably high level (Fig. 1c), but this has a cost in terms of the number of required model runs (see Fig. 2). Note that the APMC and the PMC algorithms keep  $N$  distinct particles.



**Figure 2:** Posterior quality ( $\mathbb{L}_2$ ) versus computing cost (number of simulations) averaged over 50 replicates. Vertical and horizontal bars represent the standard deviations among replicates. Algorithm parameters used for APMC:  $\alpha$  in  $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$  and  $p_{acc_{min}}$  in  $\{0.01, 0.05, 0.1, 0.2\}$ . Blue circles are used for  $p_{acc_{min}} = 0.01$ , orange triangles for  $p_{acc_{min}} = 0.05$ , green squares for  $p_{acc_{min}} = 0.1$ , and purple diamonds for  $p_{acc_{min}} = 0.2$ . PMC: red plain triangles for a sequence of tolerance levels from  $\epsilon_1 = 2$  down to  $\epsilon_{11} = 0.01$ . SMC: grey plain square for  $\alpha$  in  $\{0.9, 0.95, 0.99\}$  (from left to right),  $M = 1$  and a  $\epsilon$  target equal to 0.01. RSMC: brown plain diamond for  $\alpha = 0.5$  and a  $\epsilon$  target equal to 0.01. Results obtained with a standard rejection-based ABC algorithm are depicted with black plain circles.



**Figure 3:** (a) Boxplot of the criterion “squared  $\mathbb{L}_2$  distance times the number of simulations” for the different ABC algorithms. APMC: for  $\alpha$  in  $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$  and  $p_{acc_{min}} = 0.01$ ; SMC: for  $\alpha$  in  $\{0.9, 0.95, 0.99\}$ ,  $M = 1$  and a  $\epsilon$  target equal to 0.01; RSMC: for  $\alpha = 0.5$  and a  $\epsilon$  target equal to 0.01; ABC: for a  $\epsilon$  target equal to 0.01; PMC: for a sequence of tolerance levels from  $\epsilon_1 = 2$  to  $\epsilon_{11} = 0.01$ . (b) Criterion “squared  $\mathbb{L}_2$  distance times the number of simulations” in the APMC algorithm for the different values of  $\alpha$  and  $p_{acc_{min}}$ . Each cell depicts the average of the criterion over the 50 performed replicates of the APMC.

### Influence of parameters on APMC

The values of  $\alpha$  and  $p_{acc_{min}}$  have an impact on the studied indicators. We find that smaller  $\alpha$  and  $p_{acc_{min}}$  improve the quality of the approximation (smaller  $\mathbb{L}_2$  distance), and increase the total number of model runs, with  $p_{acc_{min}}$  having the largest effect (Fig. 2). With a large  $\alpha$ , the tolerance levels decrease slowly and there are numerous steps before the algorithm stops. In this toy example, our sim-

ulations show that all explored sets of  $(\alpha, p_{acc_{min}})$  such that  $p_{acc_{min}} < 0.1$  give good results for the criterion  $Number\ of\ simulations \times \mathbb{L}_2^2$  (Fig. 3b). Large  $\alpha$  provide slightly better results for small  $p_{acc_{min}}$  while small  $\alpha$  provide slightly better results for large  $p_{acc_{min}}$  (Fig. 3b). On this toy example it appears that intermediate values of  $\alpha$  and  $p_{acc_{min}}$  ( $0.3 \leq \alpha \leq 0.7$  and  $0.01 \leq p_{acc_{min}} \leq 0.05$ ), present a good compromise between number of model runs and the quality of the posterior approximation.

**Table 1:** SimVillages parameter descriptions

Parameters	Description	Range
$\theta_1$	Average number of children per woman	[0, 4]
$\theta_2$	Probability to accept a new residence for a household	[0, 1]
$\theta_3$	Probability to make couple for two individuals	[0, 1]
$\theta_4$	Probability to split for a couple in a year	[0, 0.5]

**Table 2:** Summary statistic descriptions

Summary statistic	Description	Measure of discrepancy
$S_1$	Number of inhabitants in 1999	$\mathbb{L}_1$ distance
$S_2$	Age distribution in 1999	$\chi^2$ distance
$S_3$	Household type distribution in 1999	$\chi^2$ distance
$S_4$	Net migration in 1999	$\mathbb{L}_1$ distance
$S_5$	Number of inhabitants in 2006	$\mathbb{L}_1$ distance
$S_6$	Age distribution in 2006	$\chi^2$ distance
$S_7$	Household type distribution in 2006	$\chi^2$ distance
$S_8$	Net migration in 2006	$\mathbb{L}_1$ distance

### Comparing performances

Whatever the value of  $\alpha$  and  $p_{acc,min}$ , the APMC algorithm always yields better results than the other three algorithms. It requires between 2 and 8 times less simulations to reach a given posterior quality  $\mathbb{L}_2$  (Fig. 2). Furthermore, good approximate posterior distributions are very quickly obtained (Fig. 2). The compromise between simulation speed and convergence level can also be illustrated using the criterion *Number of simulations*  $\times \mathbb{L}_2^2$  [17]. This criterion is smaller for the APMC algorithm (Fig. 3a).

### Application to the model SimVillages

In this section, we check if our algorithm still performs better than the PMC, the RSMC and the SMC when applied to an individual-based social model developed during the European project PRIMA[22]. The aim of the model is to simulate the effect of a scenario of job creation (or destruction) on the evolution of the population and activities in a network of municipalities.

### Model and data

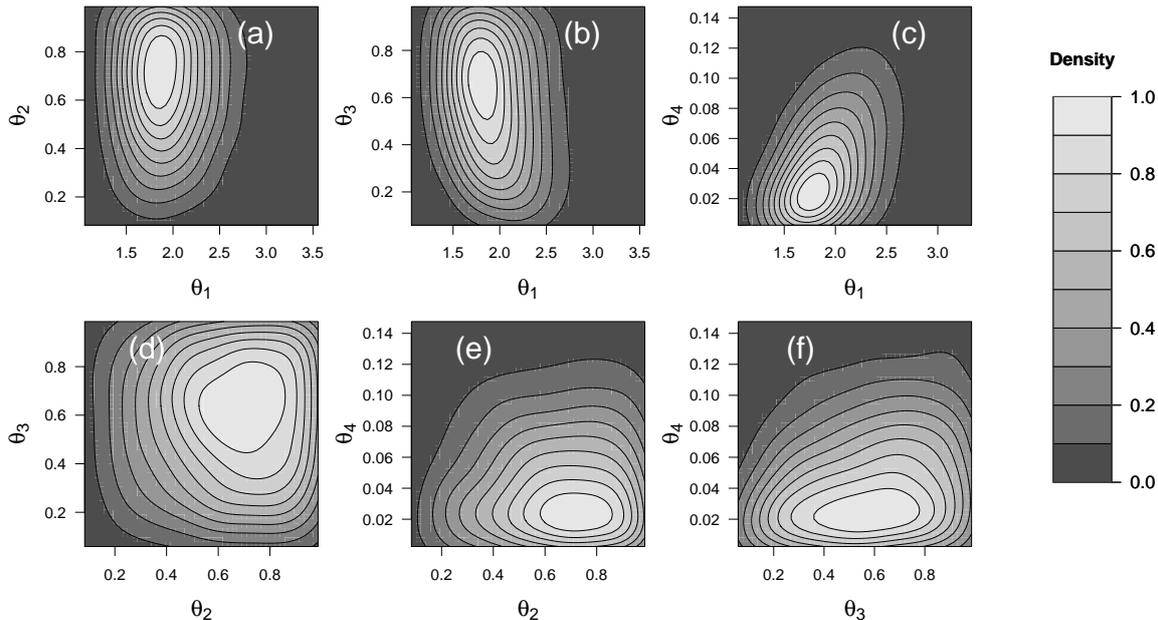
The model simulates the dynamics of virtual individuals living in 7 interconnected villages in a rural area of Auvergne (a region of Central France). A single run of the model SimVillages with seven rural municipalities takes about 1.4 seconds on a desktop

machine (PC Intel 2.83 GHz). The dynamics include demographic change (aging, marriage, divorce, births and deaths), activity change (change of jobs, unemployment, inactivity, retirement), and movings from one municipality to another or outside of the set. The model also includes a dynamics of creation / destruction of jobs of proximity services, derived from the size of the local population. More details on the model can be found in [18]. The individuals (about 3000) are initially generated using the 1990 census data of the National Institute of Statistics and Economic Studies (*INSEE*), some of them are given a job type and a location for this job (in a municipality of the set or outside), they are organised in households living in a municipality of the set. The model dynamics is mostly data driven, but four parameters cannot be directly derived from the available data. They are noted  $\theta_p$  for  $1 \leq p \leq 4$ , described in Table 1.

We use our algorithm to identify the distribution of the four parameters for which the simulations, initialized with the 1990 census data, satisfy matching criteria with the data of the 1999 and 2006 census. The set of summary statistics  $\{S_m\}_{1 \leq m \leq M}$  and the associated discrepancy measure used  $\rho_m$  are described in Table 2. We note  $S_m$  the simulated summary statistics and  $S'_m$  the observed statistics. The eight summary statistics are normalized (variance equalization) and they are combined using the infinity norm (Eq. 8):

$$\|(\rho_m(S_m, S'_m))_{1 \leq m \leq M}\|_\infty = \sup_{1 \leq m \leq M} \rho_m(S_m, S'_m) \quad (8)$$

We first generate a sample of length  $N$  from the prior  $\mathcal{U}_{[a,b]}$ , where  $[a, b]$  is available for each parameter in



**Figure 4:** Contour plot of the bivariate joint densities of  $\theta_i$  and  $\theta_j$  obtained with our algorithm, and with  $\alpha = 0.5$  and  $p_{acc_{min}} = 0.01$ ; (a)  $\theta_1$  and  $\theta_2$ ; (b)  $\theta_1$  and  $\theta_3$ ; (c)  $\theta_1$  and  $\theta_4$ ; (d)  $\theta_2$  and  $\theta_3$ ; (e)  $\theta_2$  and  $\theta_4$ ; (f)  $\theta_3$  and  $\theta_4$ .

Table 1, with a Latin hypercube [19] and we select the best  $N_\alpha$  particles. To move the particles, we use as kernel transition a multivariate normal distribution parameterized with twice the weighted variance-covariance matrix of the previous sample [20].

As in the section , we perform a parameter study and compare APMC with its three competitors. For APMC,  $\alpha$  varies in  $(\{0.3, 0.5, 0.7\})$  and  $p_{acc_{min}}$  in  $(\{0.01, 0.05, 0.1, 0.2\})$ , and we set  $N_\alpha = 5000$  particles. For the PMC, SMC and RSMC we also set  $N = 5000$  particles and a tolerance level target equal to 1.4. The tolerance value  $\epsilon = 1.4$  corresponds to the average final tolerance value we obtain with APMC for  $p_{acc_{min}} = 0.01$ . Note that otherwise this final tolerance is difficult to set properly and a worse choice for this value would have lead to worse performances of these algorithms. For the PMC algorithm, we use the decreasing sequence of tolerance levels  $\{3, 2.5, 2, 1.7, 1.4\}$ . For the SMC algorithm, we use 3 different values for the couple  $(\alpha, M)$ :  $\{(0.9, 1), (0.99, 1), (0.9, 15)\}$ . For the RSMC algorithm we use  $\alpha = 0.5$ , as in [12]. For each algorithm and parameter setting, we perform 5 replicates.

We approximated posterior density (unknown in this case) with the original rejection-based ABC algorithm, starting with  $N = 10,000,000$ , selecting 7890 particles below the tolerance level  $\epsilon = 1.4$ .

To compute the  $\mathbb{L}_2$  distance between posterior densities, we divided each parameter support into 4 equally sized bins, leading to a grid of  $4^4 = 256$  cells, and we computed on this grid the sum of the squared differences between histogram values.

## Study of APMC result

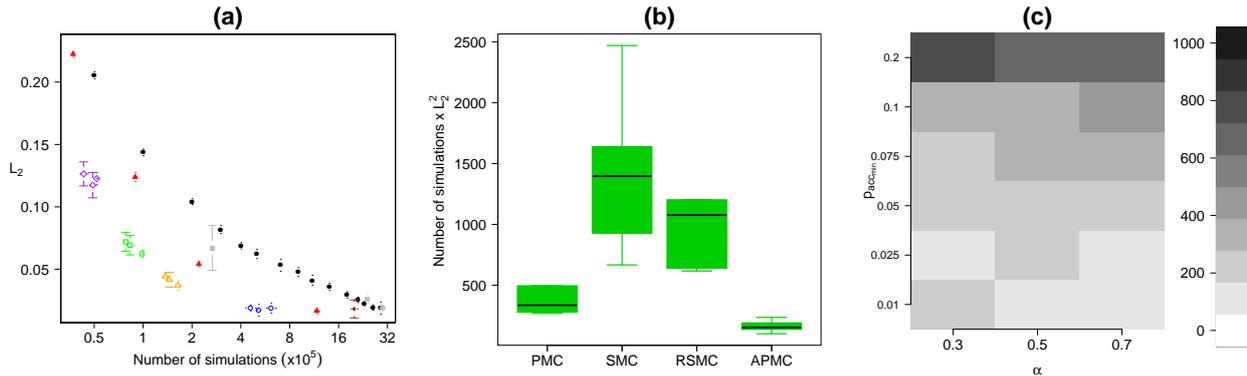
APMC yields a unimodal approximate posterior distribution for the model SimVillages (Fig. 4). Interestingly, parameters  $\theta_1$  and  $\theta_4$  are slightly correlated (Fig. 4c). This is logical since they have contradictory effects on the number of children in the population. What is less straightforward is that we are able to partly tease apart these two effects with the available census data, since we get a peak in the approximate posterior distribution instead of a ridge.

## Influence of parameters on APMC

As for the toy example, we find that the intermediate values of  $(\alpha, p_{acc_{min}})$  that we used lead to similar results (Fig. 5c). In practice, we therefore recommend to use  $\alpha = 0.5$  and  $p_{acc_{min}}$  between 0.01 and 0.05 depending on the wished level of convergence.

## Comparing performances

APMC requires between 2 and 7 times less simulations to reach a given posterior quality than the other algorithms  $\mathbb{L}_2$  (Fig. 5a). Again, the gain in simulation number is progressive during the course of the algorithm. The *Number of simulations*  $\times \mathbb{L}_2^2$  criterion is again smaller for the APMC algorithm (Fig. 5b).



**Figure 5:** (a) Posterior quality ( $\mathbb{L}_2$ ) versus computing cost (number of simulations) averaged over 5 replicates. Vertical and horizontal bars represent the standard deviations among replicates. Algorithm parameters used for APMC:  $\alpha$  in  $\{0.3, 0.5, 0.7\}$  and  $p_{acc_{min}}$  in  $\{0.01, 0.05, 0.1, 0.2\}$ . Blue circles are used for  $p_{acc_{min}} = 0.01$ , orange triangles for  $p_{acc_{min}} = 0.05$ , green squares for  $p_{acc_{min}} = 0.1$ , and purple diamonds for  $p_{acc_{min}} = 0.2$ . PMC: red plain triangles for a sequence of tolerance levels from  $\epsilon_1 = 3$  to  $\epsilon_5 = 1.4$ . SMC: grey plain square for  $(\alpha, M)$  in  $\{(0.9, 1), (0.99, 1)\}$ , grey star for  $(\alpha, M) = (0.9, 15)$  and a  $\epsilon$  target equal to 1.4. Results obtained with a standard rejection-based ABC algorithm are depicted with black plain circles. (b) Boxplot of the criterion “squared  $\mathbb{L}_2$  distance times the number of simulations” for the different algorithms. APMC: for  $\alpha$  in  $\{0.3, 0.5, 0.7\}$  and  $p_{acc_{min}} = 0.01$ ; SMC: for  $(\alpha, M)$  in  $\{(0.9, 1), (0.99, 1), (0.9, 15)\}$  and a  $\epsilon$  target equal to 0.01; RSMC: for  $\alpha = 0.5$  and a  $\epsilon$  target equal to 0.01; ABC: for a  $\epsilon$  target equal to 1.4; PMC: for a sequence of tolerance levels from  $\epsilon_1 = 3$  to  $\epsilon_5 = 1.4$ . (c) Criterion “squared  $\mathbb{L}_2$  distance times the number of simulations” in the APMC algorithm for the different values of  $\alpha$  and  $p_{acc_{min}}$ . Each cell depicts the average of the criterion over the 5 performed replicates of the APMC.

## Discussion

The good performances of APMC should of course be confirmed on other examples. Nevertheless we argue that they are due to the main assets of our approach:

- We choose an appropriate reweighting process instead of a MCMC kernel, which corrects the sampling bias without duplicating particles;
- We define an easy to interpret stopping criterion that automatically defines the number of sequential steps.

Therefore, we can have some confidence in the good performances of APMC on other examples.

In the future, it would be interesting to evaluate this algorithm on models involving a larger number of parameters and/or multi-modal posterior distributions. Moreover, APMC could benefit from other improve-

ments, in particular by performing a semi-automatic selection of informative summary statistics after the first ABC step [4, 5] and by using local regressions for post-processing the final posterior distribution [2, 3]. We did not perform such combinations in the present contribution, so that our algorithm is directly comparable with the three other sequential algorithms we looked at. However, they would be straightforward, because the different improvements concern different steps of the ABC procedure.

## Acknowledgements

This publication has been funded by the Prototypical policy impacts on multifunctional activities in rural municipalities collaborative project, European Union 7th Framework Programme (ENV 2007-1), contract no. 212345. The work of the first author has been funded by the Auvergne region.

[1] M. A. Beaumont. *Approximate Bayesian computation in evolution and ecology*, volume 41 of *Annual Review of Ecology, Evolution, and Systematics*. 2010.  
 [2] M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.  
 [3] M. G. B. Blum and O. François. Non-linear regression models for approximate Bayesian computation. *Statistics and Computing*, 20(1):63–73, 2010.  
 [4] P. Joyce and P. Marjoram. Approximately sufficient

statistics and Bayesian computation. *Statistical Applications in Genetics and Molecular Biology*, 7(1), 2008.  
 [5] P. Fearnhead and D. Prangle. Constructing summary statistics for approximate Bayesian computation: Semi-automatic ABC. Technical Report 1004.1112, arXiv.org, 2011.  
 [6] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the*

- United States of America*, 100(26):15324–15328, 2003.
- [7] D. Wegmann, C. Leuenberger, and L. Excoffier. Efficient approximate bayesian computation coupled with markov chain monte carlo without likelihood. *Genetics*, 182(4):1207–1218, 2009.
- [8] S. A. Sisson, Y. Fan, and M. M. Tanaka. Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the United States of America*, 104(6):1760–1765, 2007.
- [9] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael P. H. Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6:187, 2009.
- [10] M. A. Beaumont, J.M. Cornuet, J.M. Marin, and C. P. Robert. Adaptive approximate Bayesian computation. *Biometrika*, 96(4):983–990, 2009.
- [11] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 68(3):411–436, 2006.
- [12] C. C. Drovandi and A. N. Pettitt. Estimation of parameters for macroparasite population evolution using approximate Bayesian computation. *Biometrics*, 67(1):225–233, 2011.
- [13] P. Del Moral, A. Doucet, and A. Jasra. An adaptive sequential Monte Carlo method for approximate Bayesian computation. *Statistics and Computing*, 22(5):1009–1020, 2012.
- [14] Franck Jabot, Thierry Faure, and Nicolas Dumoulin. Easyabc: performing efficient approximate bayesian computation sampling schemes using R. *Methods in Ecology and Evolution*, 2013.
- [15] Daniel Wegmann, Christoph Leuenberger, Samuel Neuenschwander, and Laurent Excoffier. Abctoolbox: a versatile toolkit for approximate bayesian computations. *BMC Bioinformatics*, 11(1):116, 2010.
- [16] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.
- [17] P.W. Glynn and W. Whitt. The asymptotic efficiency of simulation estimators. *Oper. Res.*, 40(3):505–520, 1992.
- [18] S. Huet and G. Deffuant. Common framework for the microsimulation model in prima project. Technical report, Cemagref LISC, 2011.
- [19] R. Carnell. lhs: Latin hypercube samples. *R package version 0.5*, 2009.
- [20] S. Filippi, C. Barnes, and M. P. H. Stumpf. On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo. (arXiv:1106.6280v4), 2012.
- [21] [http://motive.cemagref.fr/people/maxime.lenormand/script\\_r\\_toyex](http://motive.cemagref.fr/people/maxime.lenormand/script_r_toyex)
- [22] PRototypical policy Impacts on Multifunctional Activities in rural municipalities - EU 7th Framework Research Programme; 2008-2011; <https://prima.cemagref.fr/the-project>

## Appendix A: Description of the algorithms

---

### Algorithm 1: Likelihood-free rejection sampler (ABC)

---

Given  $N$  the number of particles  
**for**  $i = 1$  to  $N$  **do**  
  **repeat**  
    Generate  $\theta^* \sim \pi(\theta)$   
    Simulate  $x \sim f(x|\theta^*)$   
  **until**  $\rho(S(x), S(y)) < \epsilon$   
  Set  $\theta_i = \theta^*$   
**end for**

---



---

### Algorithm 2: Population Monte Carlo Approximate Bayesian Computation (PMC)

---

Given  $N$  the number of particles and a decreasing sequence of tolerance level  $\epsilon_1 \geq \dots \geq \epsilon_T$ ,  
For  $t = 1$ ,  
**for**  $i = 1$  to  $N$  **do**  
  **repeat**  
    Simulate  $\theta_i^{(1)} \sim \pi(\theta)$  and  $x \sim f(x|\theta_i^{(1)})$   
  **until**  $\rho(S(x), S(y)) < \epsilon_1$   
  Set  $w_i^{(1)} = \frac{1}{N}$   
**end for**  
Take  $\sigma_2^2$  as twice the weighted empirical variance of  $(\theta_i^{(1)})_{1 \leq i \leq N}$   
**for**  $t = 2$  to  $T$  **do**  
  **for**  $i = 1$  to  $N$  **do**  
    **repeat**  
      Sample  $\theta_i^*$  from  $\theta_j^{(t-1)}$  with probabilities  $w_j^{(t-1)}$   
      Generate  $\theta_i^{(t)}|\theta_i^* \sim \mathcal{N}(\theta_i^*, \sigma_i^2)$  and  $x \sim f(x|\theta_i^{(t)})$   
    **until**  $\rho(S(x), S(y)) < \epsilon_t$   
    Set  $w_i^{(t)} \propto \frac{\pi(\theta_i^{(t)})}{\sum_{j=1}^N w_j^{(t-1)} \sigma_t^{-1} \varphi(\sigma_t^{-1}(\theta_i^{(t)} - \theta_j^{(t-1)}))}$   
  **end for**  
  Take  $\sigma_{t+1}^2$  as twice the weighted empirical variance of  $(\theta_i^{(t)})_{1 \leq i \leq N}$   
**end for**  
Where  $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$

---

---

**Algorithm 3:** Sequential Monte Carlo Approximate Bayesian Computation Replenishment (RSMC)
 

---

Given  $N$ ,  $\epsilon_1$ ,  $\epsilon_T$ ,  $c$ ,  $\alpha \in [0, 1]$  and  $N_\alpha = \lfloor \alpha N \rfloor$ ,  
**for**  $i = 1$  to  $N$  **do**  
   **repeat**  
     Simulate  $\theta_i \sim \pi(\theta)$  and  $x \sim f(x|\theta_i)$   
      $\rho_i = \rho(S(x), S(y))$   
   **until**  $\rho_i \leq \epsilon_1$   
**end for**  
 Sort  $(\theta_i, \rho_i)$  by  $\rho_i$   
 Set  $\epsilon_{MAX} = \rho_N$   
**while**  $\epsilon_{MAX} > \epsilon_T$  **do**  
   Remove the  $N_\alpha$  particles with largest  $\rho$   
   Set  $\epsilon_{NEXT} = \rho_{N-N_\alpha}$   
   Set  $i_{acc} = 0$   
   Compute the parameters of the proposal *MCMC*  $q(\cdot, \cdot)$  with the  $N - N_\alpha$  particles.  
   **for**  $j = 1$  to  $N_\alpha$  **do**  
     Simulate  $\theta_{N-N_\alpha+j} \sim (\theta_i)_{1 \leq i \leq N-N_\alpha}$   
     **for**  $k = 1$  to  $R$  **do**  
       Generate  $\theta^* \sim q(\theta^*, \theta_{N-N_\alpha+j})$  et  $x^* \sim f(x^*|\theta^*)$   
       Generate  $u \sim \mathcal{U}_{[0,1]}$   
       **if**  $u \leq 1 \wedge \frac{\pi(\theta^*)q(\theta_{N-N_\alpha+j}, \theta^*)}{\pi(\theta_{N-N_\alpha+j})q(\theta^*, \theta_{N-N_\alpha+j})} \mathbb{1}_{\rho(S(x^*), S(y)) \leq \epsilon_{NEXT}}$  **then**  
         Set  $\theta_{N-N_\alpha+j} = \theta^*$   
         Set  $\rho_{N-N_\alpha+j} = \rho(S(x^*), S(y))$   
          $i_{acc} \leftarrow i_{acc} + 1$   
       **end if**  
     **end for**  
   **end for**  
   Set  $p_{acc} = \frac{i_{acc}}{RN_\alpha}$   
   Set  $R = \frac{\log(c)}{\log(1 - p_{acc})}$   
**end while**

---

---

**Algorithm 4:** Adaptive Sequential Monte Carlo Approximate Bayesian Computation (SMC)
 

---

Given  $N, M, \alpha \in [0, 1], \epsilon_0 = \infty, \epsilon$  and  $N_T$ ,

For  $t = 0$ ,

**for**  $i = 1$  to  $N$  **do**

  Simulate  $\theta_i^{(0)} \sim \pi(\theta)$

**for**  $k = 1$  to  $M$  **do**

    Simulate  $X_{(i,k)}^{(0)} \sim f(\cdot | \theta_i^{(0)})$

**end for**

  Set  $W_i^{(0)} = \frac{1}{N}$

**end for**

We have  $ESS((W_i^{(0)}), \epsilon_0) = N$  where  $ESS((W_i^{(0)}), \epsilon_0) = \left( \sum_{i=1}^N (W_i^{(0)})^2 \right)^{-1}$

Set  $t = 1$

**while**  $\epsilon_{t-1} > \epsilon$  **do**

  Determine  $\epsilon_t$  resolving  $ESS((W_i^{(t)}), \epsilon_t) = \alpha ESS((W_i^{(t-1)}), \epsilon_{t-1})$  where  $W_i^{(t)} \propto W_i^{(t-1)} \frac{\sum_{k=1}^M \mathbb{1}_{A_{\epsilon_{t-1}, y}}(X_{(i,k)}^{(t-1)})}{\sum_{k=1}^M \mathbb{1}_{A_{\epsilon_{t-1}, y}}(X_{(i,k)}^{(t-1)})}$  et

$A_{\epsilon, y} = \{x | \rho(S(x), S(y)) < \epsilon\}$

**if**  $\epsilon_t < \epsilon$  **then**

$\epsilon_n = \epsilon$

**end if**

**if**  $ESS((W_i^{(t)}), \epsilon_t) < N_T$  **then**

**for**  $i = 1$  to  $N$  **do**

      Simulate  $(\theta_{(i)}^{(t-1)}, X_{(i,1:M)}^{(t-1)})$  in  $(\theta_{(j)}^{(t-1)}, X_{(j,1:M)}^{(t-1)})$  with probabilities  $W_j^{(t-1)}, 1 \leq j \leq N$

      Set  $W_i^{(t)} = \frac{1}{N}$

**end for**

**end if**

**for**  $t = 1$  to  $N$  **do**

**if**  $W_j^{(t)} > 0$  **then**

      Generate  $\theta^* \sim K(\theta^* | \theta_{(i)}^{(t-1)})$

**for**  $k = 1$  to  $M$  **do**

        Simulate  $X_{(*,k)} \sim f(\cdot | \theta^*)$

**end for**

      Generate  $u < \mathcal{U}_{[0,1]}$

**if**  $u \leq 1 \wedge \frac{\sum_{k=1}^M \mathbb{1}_{A_{\epsilon_t, y}}(X_{(*,k)}) \pi(\theta^*) K_t(\theta_{(i)}^{(t-1)} | \theta^*)}{\sum_{k=1}^M \mathbb{1}_{A_{\epsilon_t, y}}(X_{(i,k)}^{(t-1)}) \pi(\theta_{(i)}^{(t-1)}) K_t(\theta^* | \theta_{(i)}^{(t-1)})}$  **then**

        Set  $(\theta_{(i)}^{(t)}, X_{(i,1:M)}^{(t)}) = (\theta^*, X_{(*,1:M)})$

**else**

        Set  $(\theta_{(i)}^{(t)}, X_{(i,1:M)}^{(t)}) = (\theta_{(i)}^{(t-1)}, X_{(i,1:M)}^{(t-1)})$

**end if**

**end if**

**end for**

**end while**

---

---

**Algorithm 5:** Adaptive Population Monte Carlo Approximate Bayesian Computation
 

---

Given  $N$ ,  $N_\alpha = \lfloor \alpha N \rfloor$  the number of particles to keep at each iteration among the  $N$  particles ( $\alpha \in [0, 1]$ ) and  $p_{accmin}$  the minimal acceptance rate.

**for**  $t = 1$  **do**

**for**  $i = 1$  to  $N$  **do**

    Simulate  $\theta_i^{(0)} \sim \pi(\theta)$  and  $x \sim f(x|\theta_i^{(0)})$

    Set  $\rho_i^{(0)} = \rho(S(x), S(y))$

    Set  $w_i^{(0)} = 1$

**end for**

  Let  $\epsilon_1 = Q_{\rho^{(0)}}(\alpha)$  the first  $\alpha$ -quantile of  $\rho^{(0)}$  where  $\rho^{(0)} = \{\rho_i^{(0)}\}_{1 \leq i \leq N}$

  Let  $\{(\theta_i^{(1)}, w_i^{(1)}, \rho_i^{(1)})\} = \{(\theta_i^{(0)}, w_i^{(0)}, \rho_i^{(0)}) | \rho_i^{(0)} \leq \epsilon_1, 1 \leq i \leq N\}$

  Take  $\sigma_1^2$  as twice the weighted empirical variance of  $\{(\theta_i^{(1)}, w_i^{(1)})\}_{1 \leq i \leq N_\alpha}$

  Set  $p_{acc} = 1$

$t \leftarrow t + 1$

**end for**

**while**  $p_{acc} > p_{accmin}$  **do**

**for**  $i = N_\alpha + 1$  to  $N$  **do**

    Pick  $\theta_i^*$  from  $\theta_j^{(t-1)}$  with probability  $\frac{w_j^{(t-1)}}{\sum_{k=1}^{N_\alpha} w_k^{(t-1)}}$ ,  $1 \leq j \leq N_\alpha$

    Generate  $\theta_i^{(t-1)} | \theta_i^* \sim \mathcal{N}(\theta_i^*, \sigma_{t-1}^2)$  and  $x \sim f(x|\theta_i^{(t-1)})$

    Set  $\rho_i^{(t-1)} = \rho(S(x), S(y))$

    Set  $w_i^{(t-1)} = \frac{\pi(\theta_i^{(t-1)})}{\sum_{j=1}^{N_\alpha} (w_j^{(t-1)} / \sum_{k=1}^{N_\alpha} w_k^{(t-1)}) \sigma_{t-1}^{-1} \varphi(\sigma_{t-1}^{-1} (\theta_i^{(t-1)} - \theta_j^{(t-1)}))}$

**end for**

  Set  $p_{acc} = \frac{1}{N - N_\alpha} \sum_{k=N_\alpha+1}^N \mathbf{1}_{\rho_i^{(t-1)} < \epsilon_{t-1}}$

  Let  $\epsilon_t = Q_{\rho^{(t-1)}}(\alpha)$  where  $\rho^{(t-1)} = \{\rho_i^{(t-1)}\}_{1 \leq i \leq N}$

  Let  $\{(\theta_i^{(t)}, w_i^{(t)}, \rho_i^{(t)})\} = \{(\theta_i^{(t-1)}, w_i^{(t-1)}, \rho_i^{(t-1)}) | \rho_i^{(t-1)} \leq \epsilon_t, 1 \leq i \leq N\}$

  Take  $\sigma_t^2$  as twice the weighted empirical variance of  $\{(\theta_i^{(t)}, w_i^{(t)})\}_{1 \leq i \leq N_\alpha}$

$t \leftarrow t + 1$

**end while**

Where  $\forall u \in [0, 1]$  and  $X = \{x_1, \dots, x_n\}$ ,  $Q_X(u) = \inf\{x \in X | F_X(x) \geq u\}$  and  $F_X(x) = \frac{1}{n} \sum_{k=1}^n \mathbf{1}_{x_k \leq x}$ .

Where  $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$

---

### Appendix B: Proof that the algorithm stops

We know that there exists  $\epsilon_\infty > 0$  such that  $\epsilon_t \xrightarrow{t \rightarrow +\infty} \epsilon_\infty$  because, by construction of the algorithm ( $\epsilon_t$ ) is a positive decreasing sequence and it is bounded by 0.

For each  $\theta \in \Theta$ , we consider the distance  $(\rho(x, y)|\theta)$  as a random variable  $\rho(\theta)$ . Let  $f_{\rho(\theta)}$  be the probability density function of  $\rho(\theta)$ .

The probability  $\mathbb{P}[\rho(\theta) \geq \epsilon_t]$  that the drawn distance associated to parameter  $\theta$  is higher than the current tolerance  $\epsilon_t$  satisfies:

$$\begin{aligned} \mathbb{P}[\rho(\theta) \geq \epsilon_t] &= 1 - \mathbb{P}[(\rho(\theta) < \epsilon_t)] \\ &= 1 - \int_{\epsilon_\infty}^{\epsilon_t} f_{\rho(\theta)}(x) dx \end{aligned}$$

We define:

$$\mathbb{P}_{max} = \sup_{\theta \in \Theta} \left\{ \sup_{x \in \mathbb{R}^+} \{f_{\rho(\theta)}(x)\} \right\}$$

We have:

$$\mathbb{P}[\rho(\theta) \geq \epsilon_t] \geq 1 - \mathbb{P}_{max}(\epsilon_t - \epsilon_\infty)$$

The  $N - N_\alpha$  particles are independent and identically distributed from  $\pi_{t+1}$  the density defined by the algorithm, hence the probability  $\mathbb{P}[p_{acc}(t+1) = 0]$  that no particle is accepted at step  $t+1$  is such that:

$$\mathbb{P}[p_{acc}(t+1) = 0] \geq (1 - \mathbb{P}_{max}(\epsilon_t - \epsilon_\infty))^{N - N_\alpha}$$

If  $\mathbb{P}_{max} < +\infty$ , because  $\epsilon_t - \epsilon_\infty \xrightarrow{t \rightarrow +\infty} 0$ , we have:

$$\mathbb{P}[p_{acc}(t+1) = 0] \xrightarrow{t \rightarrow +\infty} 1$$

We can conclude that  $p_{acc}(t)$  converges in probability towards 0 if  $\mathbb{P}_{max} < +\infty$ . This ensures that the algorithm stops, whatever the chosen value of  $p_{acc_{min}}$ .