ARTICLE IN PRESS

Journal of Mathematical Psychology ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Journal of Mathematical Psychology

journal homepage: www.elsevier.com/locate/jmp

Approximate Bayesian computation with differential evolution☆

Brandon M. Turner[a], Per B. Sederberg[b,*]

[a] University of California, Irvine, United States
[b] The Ohio State University, United States

ARTICLE INFO

Article history:
Received 16 May 2012
Received in revised form
12 June 2012
Available online xxxx

Keywords:
Approximate Bayesian computation
Differential evolution
Computational modeling
Likelihood-free inference

ABSTRACT

Approximate Bayesian computation (ABC) is a simulation-based method for estimating the posterior distribution of the parameters of a model. The ABC approach is instrumental when a likelihood function for a model cannot be mathematically specified, or has a complicated form. Although difficulty in calculating a model's likelihood is extremely common, current ABC methods suffer from two problems that have largely prevented their mainstream adoption: long computation time and an inability to scale beyond a few parameters. We introduce differential evolution as a computationally efficient genetic algorithm for proposal generation in our ABC sampler. We show how using this method allows our new ABC algorithm, called ABCDE, to obtain accurate posterior estimates in fewer iterations than kernel-based ABC algorithms and to scale to high-dimensional parameter spaces that have proven difficult for current ABC methods.

1. Introduction

Bayesian statistics has become an enormously popular tool for the analysis of random variables. While there are a myriad of practical and theoretical reasons to adopt the Bayesian framework, one clear advantage is the quantification of uncertainty in the form of the posterior distribution. In contrast to classic null hypothesis testing, directed analysis of the posterior distribution allows for statistical inference that does not compromise the process theory behind the experiments generating the data. In the last decade, many have turned to the problem of Bayesian estimation when the likelihood function is difficult or impossible to evaluate. One popular method for solving this problem is called approximate Bayesian computation (ABC). While ABC originated in genetics (Beaumont, Zhang, & Balding, 2002; Marjoram, Molitor, Plagnol, & Tavare, 2003; Pritchard, Seielstad, Perez-Lezaun, & Feldman, 1999; Tavaré, Balding, Griffiths, & Donnelly, 1997), it has received a considerable amount of attention in ecology, epidemiology, and systems biology (Beaumont, 2010), as well as, more generally, with regard to model choice (Robert, Cornuet, Marin, & Pillai, 2011). Recently, ABC has also become an important tool in the analysis of models of human behavior, whose likelihoods are often rather complex (Turner & Van Zandt, 2012).

The basic ABC algorithm proceeds as follows. Given an observed data set Y, some unknown parameters θ with prior distribution $\pi(\theta)$, and an assumed model such that $Y \sim \text{Model}(y|\theta)$, ABC algorithms propose candidate parameter values by randomly sampling θ^* and generating a data set $X \sim \text{Model}(x|\theta^*)$, where the method used to draw θ^* varies from one algorithm to the next. Because this random data set is always generated and paired with the random parameter value θ^*, we often use the notation (θ^*, X) to refer to a single point in the joint distribution of θ^* and X. The idea is that if the distance $\rho(X, Y)$ between the two data sets is small enough, specifically, less than a value ϵ, we can assume that the proposed candidate parameter value θ^* has some nonzero probability of being in the posterior distribution $\pi(\theta|Y)$. In so doing, ABC algorithms attempt to approximate the posterior distribution by sampling from

$$\pi(\theta|Y) \propto \int_{\mathcal{X}} \pi(\theta)\, \text{Model}(x|\theta)\, I(\rho(X, Y) \leq \epsilon)\, dx, \tag{1}$$

where $\mathcal{X}$ is the support of the simulated data and $I(a)$ is an indicator function returning a one if the condition a is satisfied and a zero otherwise.

The ABC rejection framework (see Turner & Van Zandt, 2012 for a tutorial) has been embedded in Markov chain Monte Carlo (Marjoram et al., 2003), sequential Monte Carlo (Beaumont, Cornuet, Marin, & Robert, 2009; Del Moral, Doucet, & Jasra, 2008; Sisson, Fan, & Tanaka, 2007; Toni, Welch, Strelkowa, Ipsen, & Stumpf, 2009), expectation propagation (Barthelmé & Chopin, 2011), and Gibbs sampling (Turner & Van Zandt, submitted for publication). While these algorithms have become

☆ Portions of this work were presented at the 8th Annual Context and Episodic Memory Symposium, Bloomington, Indiana, and the 45th Annual Meeting of the Society for Mathematical Society, Columbus, Ohio. The authors would like to thank Trisha Van Zandt, Simon Dennis, Rich Shiffrin, Jay Myung, and Dan Navarro for helpful comments that improved an earlier version of this article.

* Corresponding author.
E-mail addresses: bmturner@uci.edu (B.M. Turner), sederberg.1@osu.edu (P.B. Sederberg).

0022-2496/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jmp.2012.06.004

increasingly more complex and efficient, they all still rely on satisfying the condition $\rho(X, Y) \leq \epsilon$. Hence, the accuracy of the estimated posterior distribution and the computational efficiency hinge on the selection of the tolerance threshold ϵ: large values of ϵ produce inaccurate posteriors while small values of ϵ produce long computation times. Long computation times are a result of the decrease in the likelihood of proposing acceptable parameter values θ^* that produce data X such that the condition $\rho(X, Y) \leq \epsilon$ is satisfied.

A number of methods have been proposed to balance the trade-off between long computation times and accurate posterior estimates. One common approach is to select a set of ϵ values that decrease monotonically from a larger tolerance value to some small one. This allows the algorithm to converge slowly to a computationally feasible final value of ϵ. Other methods allow for larger values of ϵ, and instead rely on a post-simulation correction to obtain accurate estimates of the posterior (Beaumont et al., 2002; Blum & François, 2010). These algorithms proceed as described above but then use regression techniques to reweigh the samples based on their distance from the observed data Y. Although these techniques are very useful in practice, the post hoc nature of the correction suggests that the samplers used prior to the correction could be substantially improved. That is, algorithms that use the reweighing scheme internally would be more efficient than the regression correction schemes.

Although the basic ABC framework has been instrumental in furthering the widespread usage of Bayesian analysis, it currently suffers from the so-called "curse of dimensionality" (Beaumont, 2010). That is, when the number of parameters is increased the acceptance rate declines dramatically because successful sets of parameter values (i.e., sets of parameter values that satisfy the tolerance threshold condition) become very hard to find. To attenuate this problem, a few new algorithms replace the indicator function in Eq. (1) with a kernel function $\psi(\rho(X, Y)|\delta)$ (Wilkinson, submitted for publication). The kernel function provides continuous measures to the evaluation of a proposal parameter set θ^* rather than a dichotomous accept or reject (a one or a zero, respectively). While this dramatically improves the computational efficiency, the accuracy of the estimated posterior now relies on the selection of δ. In addition, when sampling from a high-dimensional parameter space, the selection of tuning parameters such as the transition kernel becomes vitally important.

In this article we present a new ABC algorithm, which we call ABCDE, that uses differential evolution (DE) as a means of proposal generation (Hu & Tsui, 2005; ter Braak, 2006; Vrugt et al., 2009). DE is an extremely efficient population-based genetic algorithm for minimizing real-valued cost functions (Storn & Price, 1997). Instead of relying solely on random mutations of each member of the population to drive evolution, DE creates a self-organizing population by evolving each member based on the weighted difference between two other members of the population, similar in spirit to particle swarm optimization (Havangi, Nekoui, & Teshnehlab, 2010; Kennedy & Eberhart, 1995; Tong, Fang, & Xu, 2006). In practice, DE requires far fewer population members than other genetic algorithms, typically just enough to span the possible parameter space, because the entire population evolves towards the minimum cost function values (or in our case the regions of the parameter space with the highest densities). Here we merge ABC and DE, including enhancements to the basic DE algorithm to make it suitable for posterior estimation instead of simply locating the global minimum (Hu & Tsui, 2005).

We first describe the algorithm (presented in Fig. 1) and motivate the use of each of its components. We then use the algorithm to fit three models whose posterior distributions are known to assess the accuracy of the estimates obtained using

```
1: Initialize θ_{1:K,1:G,1} by sampling from the prior:
2: θ_{1:K,1:G,1} ~ π(θ)
3: for 2 ≤ t ≤ T do
4:   if p*_1 < α then
5:     θ_{1:K,1:G,t} ← Migrate(θ_{1:K,1:G,t−1})
6:   end if
7:   for 1 ≤ k ≤ K do
8:     if p*_2 < β then
9:       θ_{k,1:G,t} ← Mutate(θ_{k,1:G,t−1})
10:    else
11:      θ_{k,1:G,t} ← Crossover(θ_{k,1:G,t−1})
12:    end if
13:  end for
14: end for
```

Fig. 1. The ABCDE algorithm for estimating the posterior distribution of θ.

ABCDE. In the first example, we show that a reduced version of the ABCDE algorithm can easily handle the classic mixture of normals example that is prevalent in the ABC literature (Beaumont et al., 2009; Sisson et al., 2007; Toni et al., 2009; Wilkinson, submitted for publication). We then assess the accuracy of the ABCDE algorithm by comparing it to a basic kernel-based ABC algorithm (Wilkinson, submitted for publication). To do so, we use both algorithms to fit a model whose joint posterior distribution is highly correlated. In our final example, we test the scalability of ABCDE by fitting it to a model with 20 parameters, a problem that is extremely difficult for existing ABC algorithms to solve. We close with a brief discussion of the potential advancements and provide guidelines for the selection of the tuning parameters.

2. The algorithm

The ABCDE algorithm combines DE with a distributed genetic algorithm approach that relies on three steps: crossover, mutation, and migration (Hu & Tsui, 2005). After a high-level overview of the algorithm, we will discuss each of these steps in turn. Fig. 1 illustrates the structure of the ABCDE algorithm. We begin by selecting a pool of P particles and dividing this pool evenly into K groups, each of size $G = P/K$. Although we will refer to the group of individual members of our sampler as particles, they are actually a pool of Markov chains that interact, forming an entire system very similar to population Monte Carlo samplers. The particles are first initialized; for example, one could initialize the particles by sampling from the prior distribution $\pi(\theta)$ for the parameters θ. In Fig. 1, this step is represented by Lines 1–2. Once all particles are initialized, the first step in the algorithm is to decide whether or not a migration step should be performed with probability α. As we will soon discuss, the migration step is the distributed genetic algorithm method of diversifying each of the groups, and it is represented in Fig. 1 by Lines 4–6. Subsequently, the ABCDE algorithm updates the particles by means of a crossover step (the core DE particle update mechanism) performed with probability β or by means of a mutation step (the standard particle perturbation method) with probability $1 - \beta$. The decision to update via the crossover or the mutation step is performed for each of the K groups. This cyclical updating procedure is represented in Fig. 1 by Lines 7–13.

In practice, the "decision" tuning parameters will be small (e.g., $\alpha = \beta = 0.10$), such that a majority of the particle updates

Please cite this article in press as: Turner, B. M., & Sederberg, P. B. Approximate Bayesian computation with differential evolution. Journal of Mathematical Psychology (2012), doi:10.1016/j.jmp.2012.06.004

occur via the DE crossover step, and migration only occurs a few times throughout the simulation. The new proposed particles are assigned "fitness" values, which are used to decide individually whether the new proposals will replace the current particles. As with all ABC algorithms, an estimate of the posterior is formed by collapsing the values obtained from the sampler across all iterations, ignoring a (short) burn-in period. In the next sections, we first describe how to evaluate the particles at each iteration and then we discuss each of the components of the ABCDE algorithm in detail.

2.1. Weighing particles

One of the advancements that makes the ABCDE algorithm possible is the ability to assign continuous measures for a particle's "fitness", rather than simply accepting or rejecting a proposed parameter value. We make a careful distinction here between a particle's weight and a particle's fitness. There are many particle filtering ABC algorithms (e.g., Beaumont et al. (2009), Del Moral et al. (2008), Sisson et al. (2007), and Toni et al. (2009)) that apply continuous weights to their particles; however, these algorithms do not apply continuous measures to a particle's fitness. Instead, they rely on a simple accept/reject evaluation of a particle's fitness prior to calculating the weights. That is, when a particle is proposed that does not satisfy the tolerance condition $\rho(X, Y) \leq \epsilon$, the proposal is given a weight of zero because the indicator function in Eq. (1) will be zero. Thus, the fitness evaluation used by these rejection-based algorithms can result in many samples being drawn that are never used, producing high inefficiency.

To calculate our weights, we draw on a framework for applying continuous measures to a particle's fitness (Beaumont et al., 2002; Blum & François, 2010; Wilkinson, submitted for publication). Specifically, we assume that the data Y is a realization of a model simulation under the best possible parameter values plus some random error, so that

$$Y = \text{Model}(y|\theta) + \zeta, \tag{2}$$

where ζ follows the distribution $\psi(\cdot|\delta)$ governed by the parameter δ. By assuming that the random error is continuous, we can use this error term to evaluate a particle's fitness. For example, if $\psi(\cdot|\delta)$ is centered at zero, we can compute the distance $\rho(X, Y)$ and evaluate $\psi(\rho(X, Y)|\delta)$. If $\psi(\cdot|\delta)$ then follows, say, a normal distribution, simulated data that are closer to the observed data will have higher $\psi(\rho(X, Y)|\delta)$ values than simulated data that are far from the observed data because $\rho(X, Y)$ will be smaller. As a result, the approximation in Eq. (1) becomes

$$\pi(\theta|Y) \propto \int_{\mathcal{X}} \pi(\theta)\, \text{Model}(x|\theta)\ \psi(\rho(X, Y)|\delta)\, dx.$$

We will refer to the class of algorithms that use this weighing scheme as kernel-based ABC (KABC) algorithms. Because ABCDE is, at its core, a KABC algorithm, we will compare the efficiency of the proposal mechanism used in ABCDE (discussed in detail below) against the proposal mechanism used by a simple Markov chain Monte Carlo KABC algorithm later on in this article.

The ABCDE algorithm uses the Metropolis–Hastings probability to determine if a proposal should be accepted or rejected, and so we begin by outlining the Metropolis–Hastings step in the context of our algorithm. First, we let the current particle state at Iteration $t+1$ be denoted θ_t and the corresponding simulated data that was generated on Iteration t be denoted X_t. Then, for a given proposal and corresponding simulated data (θ^*, X), we can evaluate the proposal's fitness by calculating

$$\pi(\theta^*)\psi(\rho(X, Y)|\delta)q(\theta_t|\theta^*),$$

where $q(a|b)$ is called the "transition kernel" and it represents the probability of transitioning to a from b. We will delay further discussion of the transition kernel to the sections on the mutation and crossover steps, because the form of $q(a|b)$ will vary between these two different steps because they rely on different perturbation methods. We can then compare the proposal's fitness to the current particle state by calculating

$$\pi(\theta_t)\psi(\rho(X_t, Y)|\delta)q(\theta^*|\theta_t).$$

Once the fitness for both the current state and the proposal have been calculated, we "jump" from the current state to the proposal state with Metropolis–Hastings probability

$$\min\left(1, \frac{\pi(\theta^*)\psi(\rho(X, Y)|\delta)q(\theta_t|\theta^*)}{\pi(\theta_t)\psi(\rho(X_t, Y)|\delta)q(\theta^*|\theta_t)}\right). \tag{3}$$

Here, a jump is performed by setting $(\theta_{t+1}, X_{t+1}) = (\theta^*, X)$, otherwise the chain will not move and $(\theta_{t+1}, X_{t+1}) = (\theta_t, X_t)$.

2.1.1. Selecting δ

Most other ABC algorithms rely on a rejection sampling routine to first decide if a proposal θ^* is suitable for approximating the posterior distribution. As previously discussed, this decision is based on a comparison to a tolerance threshold ϵ so that a particle is accepted only if $\rho(X, Y) \leq \epsilon$. As such the accuracy of the estimated posterior distribution $\pi(\theta|\rho(X, Y) \leq \epsilon)$ clearly depends on the precision of ϵ. Using a kernel function to evaluate a particle's fitness does not solve this estimation problem because our approximation of the true posterior distribution $\pi(\theta|Y)$ will still depend on the value of δ.

While this might appear concerning, the difference between the two approaches lies in the specification of the model, as shown in Eq. (2). The observed data is assumed to have arisen from a particular model plus some error term ζ. As a result, we can incorporate the parameters of the distribution of ζ directly into our model. Specifically, we can treat δ as a parameter in the model and estimate it rather than (subjectively) choosing a tolerance condition. To do so, we must specify a prior distribution for δ (e.g., an exponential distribution that favors small values) and include it in the calculation of Eq. (3). At first, the values of δ will be large because the chains are unlikely to be in the high-density regions of the posterior. However, as the algorithm proceeds, smaller values of δ will carry larger weights, and, as a consequence, the estimated posterior distribution will become more accurate. In the simulations that follow, we demonstrate how freeing δ allows for a nearly automatic parameter estimation procedure.

It has been shown that under the assumption of model error (see Eq. (2)), ABC provides exact samples from the posterior distribution (Wilkinson, submitted for publication). We extend this finding by applying a weighing scheme that uses continuous measures into an entire network of Markov chains to facilitate estimation that is both accurate and fast. In particular, using a system of chains allows our ABCDE algorithm to jump across the modes of a target distribution, as well as tackle more difficult high-dimensional problems where ABC MCMC approaches are likely to suffer from high rejection rates.

2.2. Crossover

We propose an efficient method of generating candidate particles based on DE (Storn & Price, 1997; ter Braak, 2006). The central idea behind DE is to allow the particles in the set to inform the proposal process. Particles that are performing well are selected to guide other, poorer performing particles to higher-density regions. We implement DE within a distributed genetic algorithm framework, which divides the population of particles into sub-populations that evolve largely independently. This distributed sub-population approach has been shown to help

Please cite this article in press as: Turner, B. M., & Sederberg, P. B. Approximate Bayesian computation with differential evolution. Journal of Mathematical Psychology (2012), doi:10.1016/j.jmp.2012.06.004

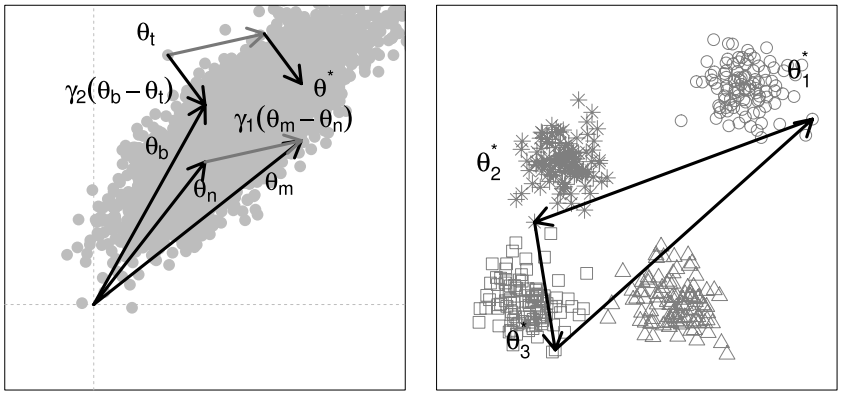

Fig. 2. Examples of the crossover step (left panel) and the migration step (right panel).

prevent falling into local minima when minimizing global cost functions (Tanese, 1989).

Fig. 2 (left panel) provides an illustration of the crossover procedure for generating the new proposals for each particle in the kth group. In this example we are updating the individual particle θ_t. First, we sample a "base" particle θ_b (shown as the vector from the origin) with the current weights as probabilities and two other particles θ_m and θ_n (bottom most vectors) with uniform probabilities, so that $\theta_m \neq \theta_n \neq \theta_t$. Next, we compute two directional vectors. The first directional vector is obtained by taking the difference between the two randomly selected particles and scaling by a constant, given by $\gamma_1(\theta_m - \theta_n)$. The dark gray lines in the left panel of Fig. 2 represent the difference vector $\theta_m - \theta_n$. This first vector is the component in ABCDE that allows us to generate efficient proposals that match the shape of the target distribution. The second directional vector is obtained by taking the difference between the particle being updated and the base particle, and scaling by another constant, given by $\gamma_2(\theta_b - \theta_t)$. The second vector is what quickly drives the ABCDE sampler to the high-density regions of the posterior. Finally, we generate a proposal θ^* for the original particle θ_t by calculating

$$\theta^* = \theta_t + \gamma_1(\theta_m - \theta_n) + \gamma_2(\theta_b - \theta_t) + b, \tag{4}$$

where $\gamma \sim CU[0.5, 1.0]$ and b is drawn from some symmetric distribution with very small variance (e.g., $b \sim CU[-\epsilon, \epsilon]$, where $\epsilon = 0.001$). The small amount of random noise introduced by sampling γ provides enough diversity in the proposals to allow the sampler to traverse the entire range of the target posterior.

Some model-fitting problems may benefit from only updating some of the parameters of each particle to slow the exploration of the parameter space (Storn & Price, 1997). To implement this, we reset some of the proposal parameters to their previous values with probability $(1 - \kappa)$. In addition to slowing the sampler, resetting only some of the parameters in the vector θ^* will increase the diversity in the pool because the new proposal will be a mixture of parameters accepted in the past and the present, which will likely make this vector very unique. In practice, κ is typically set to 1.0 or 0.9, meaning all, or nearly all, parameters are updated to match the proposal vector. The proposal θ^* is then evaluated by generating a data set $X \sim \text{Model}(x|\theta^*)$, comparing it to the observed data Y through the distance metric $\rho(X, Y)$, and then deciding to jump from the current state (θ_t, X_t) to the proposal state (θ^*, X) by evaluating Eq. (3).

2.3. The two modes of sampling

While generating proposals using Eq. (4) leads to fast convergence to the high-density regions of the posterior (i.e., the maximum a posteriori estimate), the term $\gamma_2(\theta_b - \theta_t)$ makes the probability of transitioning from θ^* to θ_t lower than the reverse move that transitions θ_t to θ^*. To see this, suppose the density of b is an independent multivariate normal centered at zero with a variance matrix of ϵ. Then, the probability of transitioning from the current state to the proposal is

$$q(\theta^*|\theta_t) = \sum_i \sum_j \sum_k w_k \mathcal{N}_p(\theta_t + \gamma_1(\theta_i - \theta_j) + \gamma_2(\theta_k - \theta_t), \epsilon),$$

where $\mathcal{N}_p(a, b)$ is the multivariate normal density of dimension equal to the number of parameters p with mean a and covariance matrix equal to the identity matrix times b, w_k is the weight associated with the kth particle θ_k and $\theta_i \neq \theta_j \neq \theta_t$. However, the reverse move is given by

$$q(\theta_t|\theta^*) = \sum_i \sum_j \sum_k w_k \mathcal{N}_p \left(\frac{\theta^* - \gamma_1(\theta_i - \theta_j) - \gamma_2(\theta_k - \theta_t)}{(1 - \gamma_2)}, \frac{\epsilon}{(1 - \gamma_2)} \right).$$

Here, we can see that $q(\theta^*|\theta_t) \neq q(\theta_t|\theta^*)$, but one could calculate the transition probabilities for each possible state to evaluate Eq. (3). Unfortunately, calculation of these transition probabilities would need to be done on each iteration, which can be very time consuming when the population of particles G is large.

However, when $\gamma_2 = 0$, Eq. (4) reduces to

$$\theta^* = \theta_t + \gamma_1(\theta_m - \theta_n) + b, \tag{5}$$

and the probability of transitioning from θ_t to θ^* is given by

$$q(\theta^*|\theta_t) = \sum_i \sum_j \mathcal{N}_p(\theta_t + \gamma_1(\theta_i - \theta_j), \epsilon),$$

which will be equal to the probability of the reverse move given by

$$q(\theta_t|\theta^*) = \sum_i \sum_j \mathcal{N}_p(\theta^* - \gamma_1(\theta_i - \theta_j), \epsilon) = \sum_i \sum_j \mathcal{N}_p(\theta^* + \gamma_1(\theta_j - \theta_i), \epsilon),$$

because the elements of the difference vector are selected with equal probability.

As a result of the asymmetry of $q(\cdot|\cdot)$ when $\gamma_2 \neq 0$, we implement the algorithm in two modes. The first mode is a "burn-in mode", in which γ_2 can be tuned to locate the high-density regions of the posterior. During this phase, we still evaluate Eq. (3), but we ignore the transition kernel probabilities. That is, we accept the proposal θ^* with probability

$$\min\left(1, \frac{\pi(\theta^*)\psi(\rho(X, Y)|\delta)}{\pi(\theta_t)\psi(\rho(X_t, Y)|\delta)}\right). \tag{6}$$

Please cite this article in press as: Turner, B. M., & Sederberg, P. B. Approximate Bayesian computation with differential evolution. Journal of Mathematical Psychology (2012), doi:10.1016/j.jmp.2012.06.004

This approach is very similar to methods of generating initial values for chains used when carrying out Markov chain Monte Carlo sampling. One could even adopt stricter, deterministic rules for proposal acceptance such as accepting a proposal θ^* if

$$\pi(\theta^*)\psi(\rho(X, Y)|\delta) > \pi(\theta_t)\psi(\rho(X_t, Y)|\delta). \quad (7)$$

However, we recommend retaining the random acceptance rule in Eq. (3) because it results in more variability in the particle set. Once the high density regions have been found, we switch to "sampling mode", in which we constrain $\gamma_2 = 0$ and fix δ to the value obtained during the burn-in mode. The estimate of δ can be any user-defined value, but we recommend using the median or the minimum of the estimates from the pool of particles. As explained, setting $\gamma_2 = 0$ results in $q(\theta^*|\theta_t) = q(\theta_t|\theta^*)$ and Eq. (3) can be evaluated without having to calculate the transition probabilities $q(\theta^*|\theta_t)$ and $q(\theta_t|\theta^*)$ because they cancel out. Thus, for both the burn-in and sampling modes in the crossover step, the transition kernel can be ignored as proposals are accepted with probability given by Eq. (6).

It is important to note that accepting new particles based on the Metropolis–Hastings probability ensures that the distribution of particles in a group will match the target distribution. While proposals with higher target densities are always accepted, proposals with lower target densities can also be accepted (see Eq. (6), and, more generally, Eq. (3)), so the sampler will maintain the diversity required to explore the entire target distribution. However, if we only accepted new proposals based on Eq. (7), as in standard genetic algorithms, the group of particles would converge to the maximum a posteriori probability estimate and it would not match the variability of the true target distribution. We stress that while using Eq. (7) may satisfy the goals of some researchers who seek "best-fitting" parameter values, using Eq. (7) will be unacceptable for those interested in obtaining the Bayesian posterior.

2.4. Mutation

The mutation step is very similar to other perturbation methods, such as in standard KABC algorithms (Wilkinson, submitted for publication). However, for our algorithm, we do not need our perturbation kernel to be as variable as in other ABC algorithms because we are relying on the crossover step to exhaustively search the parameter space. The mutation step serves only to further diversify the particles within the groups and, as we show in our simulations, often is not required to arrive at highly accurate posterior estimates.

The mutation step is performed for all particles in each of the K groups (see the algorithm in Fig. 1), taken in turn. On each update, a given particle, denoted θ^{**}, is perturbed using a transition kernel such that $\theta^* \sim \mathcal{K}(\theta^{**})$. For example, the transition kernel could be a normal distribution with variance equal to one so that $\theta^* \sim \mathcal{N}(\theta^{**}, 1)$. We then generate data X by using the assumed model so that $X \sim \text{Model}(x|\theta^*)$. The simulated data X is then compared to the observed data Y through the distance metric $\rho(X, Y)$, and we jump from the current state (θ_t, X_t) to the proposed state (θ^*, X) with Metropolis–Hastings probability given by Eq. (3).

For the mutation step, the transition kernel *densities* $q(\cdot|\cdot)$ are determined by the specific transition kernel $\mathcal{K}(\cdot)$ that is chosen. In the example above, the transition kernel was normal, and so the transition kernel density can be written as

$$q(a|b, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(a-b)^2}{2\sigma^2}\right],$$

where σ is a tuning parameter (e.g., $\sigma = 1$ in the example above). When the transition kernel densities are equal so that $q(a|b) = q(b|a)$, which happens when $\mathcal{K}(\cdot)$ is chosen to be symmetric, the densities cancel out and so Eq. (3) reduces to Eq. (6). However, sometimes the mutation step can benefit from asymmetric transition kernels, and so Eq. (3) is required. In particular, we often use asymmetric transition kernels when the support of a parameter does not have infinite support to improve the acceptance rate.

2.5. Migration

The migration step is the algorithm's way of diversifying particles *across* groups in a distributed genetic algorithm framework (Hu & Tsui, 2005; Tanese, 1989). Because the particles from each group are randomly traversing the space, one group may land in a high-density region of the posterior very quickly, while others converge much more slowly. To allow for efficient posterior mapping, we simply allow the poorly-performing group to be assisted by the high-performance group by initiating a particle swap between the groups. For ABCDE, the particle swap is (typically) performed on the particles that have small weights through sampling. These particles, while doing poorly in their groups, can still carry valuable information about the location of a higher-density region. Performing the swap in this way allows particles that would have been discarded (because they were the worst performing particles in their groups) a chance to become the best performing particles in another group. In this way, the "leaders" from the high-performance groups can continue directing their group to higher density regions.

To perform the migration step, we first determine the number of groups that will be involved in the swapping by sampling a number η uniformly from the set $\mathcal{K} = \{1, 2, \ldots, K\}$. Then, to determine which of the groups to use, we sample η numbers without replacement from $\mathcal{K}$, forming the group set $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_\eta\}$. Next, for each group in $\mathcal{G}$ we sample a single particle θ^* with probabilities based on the inverse of the current weight set from each group. Finally, we swap the particles from each of the sets in a cyclical fashion so that

$$\left\{\theta^*_{\mathcal{G}_1}, \theta^*_{\mathcal{G}_2}, \ldots, \theta^*_{\mathcal{G}_{\eta-1}} \theta^*_{\mathcal{G}_\eta}\right\} \rightarrow \left\{\theta^*_{\mathcal{G}_\eta}, \theta^*_{\mathcal{G}_1}, \ldots, \theta^*_{\mathcal{G}_{\eta-2}}, \theta^*_{\mathcal{G}_{\eta-1}}\right\}.$$

Unlike the mutation and crossover steps, we recommend that the migration step be deterministic, and so it will not rely on the Metropolis–Hastings probability in Eq. (3). However, probabilistic rules can be adopted so that each swap is determined by first adding a small amount of random noise and evaluating Eq. (3) as proposed by Hu and Tsui (2005). Unfortunately, in the ABC context, generating proposals in this way requires another data simulation step which might be costly, so we recommend the simpler, deterministic swapping rule.

Fig. 2 (right panel) provides an illustration of a migration step. Each of the four different groups is represented by a different symbol (circles, asterisks, squares and triangles). In this example, we randomly sampled from $\mathcal{K}$ and obtained $\eta = 3$. We then sampled from $\mathcal{K}$ again without replacement to form the set $\mathcal{G}$, consisting of the circle, asterisk and square groups. A single particle is sampled from each of the three groups, and these particles are swapped in a cyclical fashion.

3. Simulations

We now present three examples meant to test the validity, speed, and scalability of the ABCDE algorithm. First, we use the ABCDE algorithm to fit the classic mixture of normals example. In this first example, we employ a simplified version of the ABCDE algorithm that uses only the crossover step. In the second example, we compare the speed of the ABCDE algorithm to a recently developed KABC algorithm (Wilkinson, submitted for

Please cite this article in press as: Turner, B. M., & Sederberg, P. B. Approximate Bayesian computation with differential evolution. Journal of Mathematical Psychology (2012), doi:10.1016/j.jmp.2012.06.004

publication). In this example, the joint posterior distribution is highly correlated, which makes optimal transition kernels difficult to specify for other samplers (see Turner, Sederberg, Brown, and Steyvers (submitted for publication)). Finally, we use ABCDE to fit a 20-parameter model to data and show that despite the model's complexity, the ABCDE algorithm is able to obtain accurate estimates of the posterior distribution in a very small amount of time.

3.1. Mixture of normals

To demonstrate the advancement of the vectorized perturbation approach, we begin by fitting what has become a classic problem in the ABC literature, a relatively simple mixture of normal distributions (Beaumont et al., 2009; Sisson et al., 2007; Toni et al., 2009; Wilkinson, submitted for publication). We will use the ABCDE algorithm to estimate this posterior distribution using only the crossover step.

Suppose the posterior distribution of interest is given by the sum of two normal distributions with known variance and common mean parameter so that

$$f(\theta) = \frac{1}{2}\phi(\theta|0, 0.1) + \frac{1}{2}\phi(\theta|0, 1),$$

where $\phi(a, b)$ is the normal density with mean a and standard deviation b. To fit the model using ABCDE, we need only sample a candidate parameter θ and generate a single data point X from the above model. To compare the simulated data to the observed data $Y = 0$, we use a simple Euclidean metric

$$\rho(X, Y) = X - Y.$$

To estimate the posterior using ABCDE, we assumed that the distribution of errors were Gaussian so that

$$\zeta \sim \mathcal{N}(0, \delta).$$

Thus, the particle's fitness is determined by evaluating

$$\psi(\rho(X, Y)|\delta) = \phi(\rho(X, Y)|0, \delta).$$

We will again assume a noninformative prior for θ as in Beaumont et al. (2009) and Sisson et al. (2007), namely

$$\theta \sim CU[-10, 10],$$

where $CU[a, b]$ is the continuous uniform distribution over the interval $[a, b]$. In addition, we placed an exponential prior on δ so that

$$\delta \sim \text{Exp}(20).$$

We used only the crossover component of the ABCDE algorithm to fit the data so that $\alpha = \beta = 0$. We specified uniform additive sampling error so that $b \sim CU[-0.001, 0.001]$. We set $\kappa = 1$, $\gamma_2 = 0$ and sampled a new $\gamma_1 \sim CU[0.5, 1]$ for each iteration. We divided 100 total particles into 10 groups of 10 particles each. We then ran the algorithm for 500 iterations with no burn-in period, fitting both θ and δ simultaneously. The total number of model simulations required to fit the joint distribution of (θ, δ) was 50,000. Given that we were sampling from a two-dimensional space, this is substantially better than the ABC PRC algorithm,[1] which used 75,895 model simulations (Sisson et al., 2007). We emphasize that we were not able to control for the accuracy of the two samplers, so we instead based our estimation accuracy on the bottom right panel of Fig. 2 in Sisson et al. (2007).

The left panel of Fig. 3 shows the estimated target distribution obtained using the crossover component of the ABCDE algorithm along with the true target density (black line). The right panel shows the estimated posterior distribution of the standard deviation of the error term δ. The figure shows that although small values of δ are preferred (i.e., there is more density at lower values), the distribution has a long tail, extending out to around 0.4. Thus, the marginal posterior distribution of δ informs our understanding about of the model specified in Eq. (2). Specifically, the data that were observed could be the result of an appropriately selected model (i.e., the mixture of normal distributions), in which case δ would be small (e.g., $\delta = 0.05$), or it could be the result of a poorly specified model and random error, in which case δ would be larger (e.g., $\delta = 0.35$). Comparing the true density with the estimates we obtain (left panel), we see that using only the crossover component of ABCDE, we can recover the target distribution with remarkable accuracy.

This simulation was useful in demonstrating that the crossover perturbation method that relies on vector operations is equally capable of recovering the full target density of the classic mixture of normals problem, which has been challenging for previous ABC algorithms (Beaumont et al., 2009; Sisson et al., 2007). We emphasize that all that was required to fit the model was the prior specification of δ, and we did not require explicit specification of a tolerance threshold.

3.2. The Wald distribution

The analysis of response time (RT) has a long and interesting history in psychology. The focus on the RT distribution has highlighted the importance of Bayesian modeling of RTs. However, our most sophisticated models of RTs do not lend themselves well to Bayesian modeling because of their complexity (Lee, Fuss, & Navarro, 2006; Oravecz, Tuerlinckx, & Vandekerckhove, 2009; Vandekerckhove, Tuerlinckx, & Lee, 2011). Instead, Bayesian models of RT have often resorted to simplified but well-fitting descriptive models with likelihoods such as the Weibull, ex-Gaussian, or log Gaussian (Craigmile, Peruggia, & Van Zandt, 2010; Lee & Wagenmakers, 2012; Peruggia, Van Zandt, & Chen, 2002; Rouder, Lu, Speckman, Sun, & Jiang, 2005; Rouder, Sun, Speckman, Lu, & Zhou, 2003).

One statistical model of RTs that is both simple and theoretically useful is the Wald or inverse Gaussian distribution (Chhikara & Folks, 1989; Wald, 1947). The Wald distribution describes the behavior of the first-passage time of a diffusion process with a single boundary. This process can explain effects on RTs from a simple RT experiment (Luce, 1986), where subjects respond as soon as a signal is perceived, regardless of what the signal is, or go/no-go tasks (Heathcote, 2004; Schwarz, 2001). The two-parameter Wald distribution, with density

$$f(y|\alpha, \nu) = \alpha\left(2\pi y^3\right)^{-1/2}\exp\left[-\frac{(\alpha - \nu y)^2}{2y}\right], \quad (8)$$

provides a simple, psychologically meaningful interpretation of observed RT data. Specifically, the parameters α and ν correspond to different components of the psychological process of stimulus detection. The parameter α is the threshold amount of evidence that a subject requires to respond that a signal has been presented. The parameter ν is the drift rate or rate of accumulation of evidence (Heathcote, 2004; Matzke & Wagenmakers, 2009). Given these parameters, the Wald distribution can be thought of as a simplified version of the drift diffusion model (DDM; Ratcliff (1978)), which is a popular model for two-choice RT data. However, the likelihood function for the DDM is technically intractable because it requires

[1] Although the ABC PRC method was subsequently found to have a bias in its parameter estimates, this bias would not give rise to a decrease in performance for this simulation and, consequently, this comparison in total number of function evaluations is valid.

Please cite this article in press as: Turner, B. M., & Sederberg, P. B. Approximate Bayesian computation with differential evolution. Journal of Mathematical Psychology (2012), doi:10.1016/j.jmp.2012.06.004

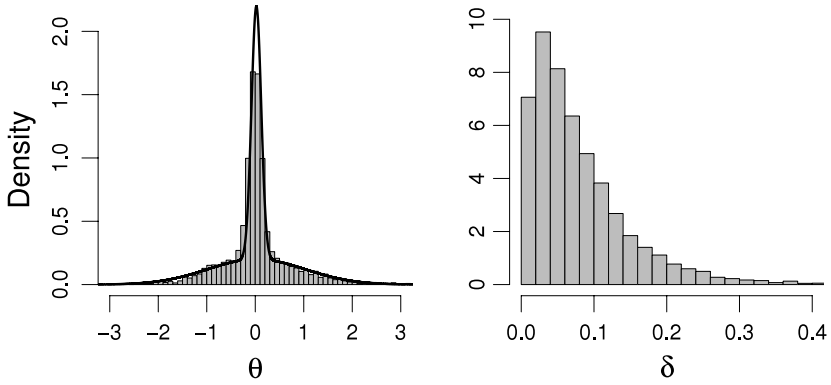

Fig. 3. The estimated target distribution (left panel) overlaid with the true target distribution (black line) and the estimated posterior distribution of the error parameter δ.

the evaluation of an oscillating but convergent infinite sum (see Feller (1968), Lee et al. (2006), and Navarro and Fuss (2009)). This is not the case for the two-parameter Wald distribution, whose likelihood function is given by

$$L(\alpha, \nu|Y) = \prod_{i=1}^{N} f(y_i|\alpha, \nu),$$

and can be factored into

$$\begin{aligned} L(\alpha, \nu|Y) &= \prod_{i=1}^{N} \alpha \left(2\pi Y_i^3\right)^{-1/2} \exp\left[-\sum_{i=1}^{N} \frac{(\alpha - \nu Y_i)^2}{2Y_i}\right] \\ &= \prod_{i=1}^{N} \alpha \left(2\pi Y_i^3\right)^{-1/2} \\ &\quad \times \exp\left[-\sum_{i=1}^{N} \frac{\alpha^2 - 2\alpha\nu Y_i + \nu Y_i^2}{2Y_i}\right] \\ &= \prod_{i=1}^{N} \alpha \left(2\pi Y_i^3\right)^{-1/2} \\ &\quad \times \exp\left(-\frac{\alpha^2}{2}\sum_{i=1}^{N}\frac{1}{Y_i} - \frac{\nu^2}{2}\sum_{i=1}^{N} Y_i + N\alpha\nu\right). \end{aligned}$$

Thus, by the Fisher–Neyman Factorization Theorem (Rice, 2007), the statistics $S_1(y) = \sum_i Y_i$ and $S_2(y) = \sum_i 1/Y_i$ are jointly sufficient for the parameters α and ν.[2]

For the purposes of this demonstration, we simulated 100 RTs in a simple RT task. We denote Y_i for the ith RT and let Y_i be distributed as a Wald random variable with parameters $\{\alpha, \nu\}$. We chose mildly informative priors for each of these parameters, namely

$$\alpha \sim \Gamma(1, 1), \quad \text{and}$$
$$\nu \sim \Gamma(1, 1).$$

Our goal is to show that even without the Wald density function, ABCDE is able to recover the same posterior distributions for the parameters of this model as a standard Bayesian approach. To make the comparison between the estimates of the posteriors from different algorithms, we used three methods. The first method made use of the likelihood function and we performed DE-MCMC sampling as described in ter Braak (2006).

The other two methods were likelihood-free methods: the first was a basic KABC algorithm as presented in Wilkinson (submitted for publication), and the second was the ABCDE algorithm. To compare the algorithms as closely as possible, we constrained the ABCDE algorithm to match the KABC algorithm so that the two methods differed only in their method of proposal generation. For both of these likelihood free methods, we used the two sufficient statistics $S_1(y) = \sum_i Y_i$ and $S_2(y) = \sum_i 1/Y_i$ derived above and assumed separate error terms for each statistic. We employed a Gaussian kernel (see Silverman (1986)) and assumed that our data Y arose from the process

$$Y = \text{Model}(y|\theta) + \zeta,$$

where $\zeta = [\zeta_1, \zeta_2]$ such that $\zeta_1 \sim \mathcal{N}(0, \delta_1)$ and $\zeta_2 \sim \mathcal{N}(0, \delta_2)$. Here, ζ_1 and ζ_2 are the error terms corresponding to each sufficient statistic $S_1(y)$ and $S_2(y)$. For both likelihood-free methods, we fixed $\delta_1 = 0.005$ and $\delta_2 = 0.01$. To compare the simulated data X to the observed data Y, we used the Euclidean distance between the ith summary statistic, so that

$$\rho_i(X, Y) = S_i(X) - S_i(Y).$$

To implement the restricted ABCDE algorithm, we again set $\alpha = \beta = 0$. We used only 24 particles in a single group, to facilitate comparison across sampling methods. For this simulation, we did not use the burn-in mode ($\gamma_2 = 0$) and ran the algorithm in sampling mode for 10,000 iterations. We specified uniform additive sampling error so that $b \sim CU[-0.001, 0.001]$, a uniform distribution for $\gamma_1 \sim CU[0.5, 1]$, and set $\kappa = 1$.

To implement the KABC algorithm (Wilkinson, submitted for publication), we applied the same basic settings that were used for ABCDE and ran 24 chains independently for 10,000 iterations. To form our estimates, we discarded the first 100 iterations in both samplers. To generate proposals, we used a Gaussian transition kernel centered at the current state of a given chain with standard deviation equal to 0.5.

Fig. 4 shows the estimated marginal posteriors for each of the methods used. The top panels show the estimates obtained using ABCDE whereas the bottom panels show the estimates obtained using KABC. In each of these three panels, the true values used to generate the data are shown by vertical lines and the likelihood-informed estimates, which were obtained using DE-MCMC, are represented as the black densities. Comparing the top and bottom panels, we can see that both likelihood-free methods provide reasonable approximations to the true posterior distribution. However, there is evidence to suggest that the ABCDE algorithm has achieved a better fit. In particular, the bottom panel of Fig. 4 shows that more sampling error is present in the estimates when using KABC. The estimates do not conform well to the true posterior distributions and several outliers are present, despite having been initialized at the same locations as in ABCDE and discarding the same number of burn-in samples.

[2] This result is easy to see because the Wald distribution belongs to the exponential family, so sufficient statistics are the functions of the data in the exponential.

Please cite this article in press as: Turner, B. M., & Sederberg, P. B. Approximate Bayesian computation with differential evolution. Journal of Mathematical Psychology (2012), doi:10.1016/j.jmp.2012.06.004

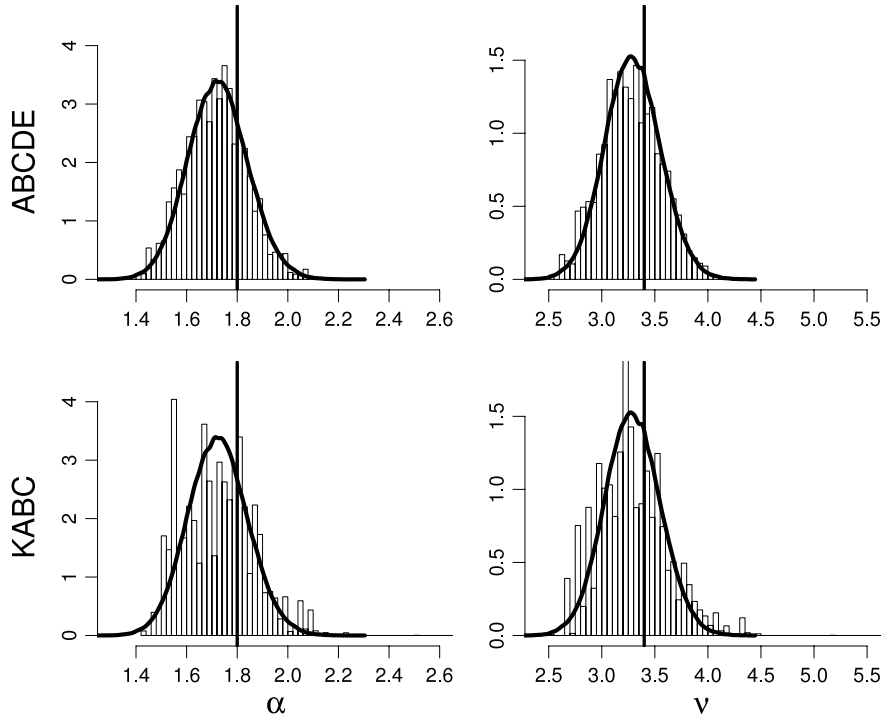

Fig. 4. The estimated marginal posterior distributions of α (left panels) and ν (right panels) using ABCDE (top panel histograms) and KABC (bottom panel histograms). The likelihood-informed estimates using DE-MCMC are shown as the black densities in all panels and the vertical lines show the parameter values used to generate the data.

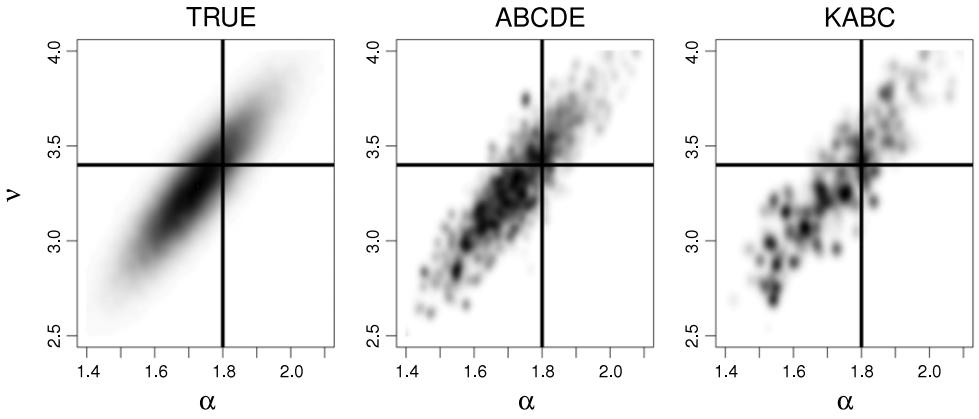

Fig. 5. The estimated joint posterior distributions of α and ν using likelihood-based DE-MCMC sampling (left panel) and likelihood-free ABCDE (middle panel) and KABC (right panel). The vertical and horizontal lines represent the parameter values used to generate the data.

Fig. 5 shows the estimated joint posterior distributions using DE-MCMC (left panel), ABCDE (middle panel), and KABC (right panel) along with the parameter values used to generate the data (red lines). The figure shows that when the likelihood is known, the sampler mixes very nicely, creating a smooth estimate. However, the two likelihood-free methods show that the chains stick for several iterations, creating black dots throughout the estimated posterior. The problem of chain sticking is prevalent in the ABC setting, especially when δ (or ϵ) is small. However, the figure shows that the estimates obtained using ABCDE fill a larger region (i.e., the estimates are much smoother) than the estimates obtained using KABC.

The performance of the two likelihood-free methods is favorable, but is to be expected given that we were able to derive sufficient statistics for the model parameters. However, showing that we can match the true posteriors when a likelihood is available is useful in illustrating that it is also possible to obtain the true posterior estimates when the likelihood is intractable, such as in the DDM. As is normal in the ABC setting, both algorithms suffered from substantial rejection rates: the ABCDE algorithm accepted 1.31% of its proposals while the KABC algorithm accepted only 0.24% of its proposals, after the burn-in period. Taken together, Figs. 4 and 5 show that this difference – a 454% improvement over the KABC algorithm – has a drastic impact on the estimates we obtain.

3.3. Bivariate normal example

As a final example, we consider the problem of scalability. Currently, ABC algorithms become extremely inefficient in high-dimensional parameter spaces, a direct result of the accept/reject nature of these algorithms. In high-dimensional spaces, proposing a successful candidate parameter vector using only a mutation kernel becomes a very improbable event.

The ABCDE algorithm, on the other hand, can be used to sample efficiently from the posterior distributions of high-dimensional models. Rather than regularly rejecting parameter vectors in low-density regions, the ABCDE algorithm will simply assign a small

Please cite this article in press as: Turner, B. M., & Sederberg, P. B. Approximate Bayesian computation with differential evolution. Journal of Mathematical Psychology (2012), doi:10.1016/j.jmp.2012.06.004

weight to the particle as measured by the kernel function. The crossover process will then evolve the population towards higher-density regions, narrowing in on the true posterior. For this next example we illustrate how well this process scales to multiple dimensions by applying the ABCDE algorithm to a set of 10 bivariate normal distributions – a 20-parameter problem – and performing inference on the parameters jointly.

We denote the observed data from the jth source as Y_j, each of which are assumed to be independently distributed from a bivariate normal distribution with an unknown mean vector μ_j and known variance matrix Σ, or

$$Y_j \sim \mathcal{N}_2\left(\mu_j = \begin{bmatrix}\mu_{j,1}\\ \mu_{j,2}\end{bmatrix}, \Sigma = \begin{bmatrix}0.01^2 & 0\\ 0 & 0.01^2\end{bmatrix}\right). \tag{9}$$

To generate the observed data for testing the model, we randomly sampled 10 points from the space $\mu_j \sim CU[0, 10] \times CU[0, 10]$, and then treated the resulting μ_js as the sole observation of the data. These sampled μ_js are shown in Fig. 6 by the "x" symbols. When simulating data X, we generated $n = 50$ observations from each bivariate normal distribution using Eq. (9) with the μ_js specified by the 20-dimensional parameter vector of the particle.

For the discriminant function, we calculated the root mean squared difference (RMSD) between the average over these 50 simulated observations $\bar{X}_j$ and the "true" observed data $\bar{Y}_j$, such that

$$\rho(X, Y) = \sqrt{\frac{1}{20}\sum_{j=1}^{20}[(\bar{X}_j - \bar{Y}_j)^2]},$$

where in this case j is indexed over the 20 parameters of the flattened 10×2 matrix of bivariate means. We then used the RMSD to compute a final weight of each proposal,

$$\psi(\rho(X, Y)|\delta) = \phi(\rho(X, Y)|0, \delta),$$

where $\phi(a, b)$ is the normal density with mean a and standard deviation b.

For each μ_j, we specified a noninformative prior over the entire space $[0, 10] \times [0, 10]$ so that

$$\mu_j \sim CU[0, 10] \times CU[0, 10].$$

In this high-dimensional space, the specification of δ becomes a difficult problem. At first, the values of $\rho(X, Y)$ will all be large, and so a small value for δ will result in the numerator of Eq. (3) being equal to zero which will lead to the rejection of the proposal. To work around this problem, we treated δ as a free parameter in the model, and assigned it a strong prior so that

$$\delta \sim \text{Exp}(20).$$

This exponential prior thus favors particles in high-density regions with small δs, but it allows for particles in the low-density regions as well (i.e., they have nonzero weights).

We set the mutation probability to $\beta = 0.0$ (i.e., we used crossover only) and crossover parameter update probability $\kappa = 0.9$ (allowing for more diversity in the crossover step). We used only one group ($G = 1$) of size $K = 50$, for a total of 50 particles. There was no need to run the migration step, so we set $\alpha = 0.0$. We again implemented burn-in and sampling periods. During the burn-in period, we sampled $\gamma_2 \sim CU[0.5, 1]$ for each update and ran the algorithm for 200 iterations. During the sampling period, we set $\gamma_2 = 0$ and ran the algorithm for 300 iterations. For both modes, we sampled $\gamma_1 \sim CU[0.5, 1]$ and $b \sim CU[-0.001, 0.001]$. To transition from the burn-in mode to the sampling mode, we initialized the chains during the sampling mode to the final values obtained during the burn-in mode. Finally, during the burn-in mode, δ was treated as a free parameter, but, during the sampling mode, δ was fixed to the minimum of the distribution obtained by the end of the burn-in mode.

For such a high-dimensional problem, one could certainly benefit from dividing the sampling procedure into blocks (Gelman, Carlin, Stern, & Rubin, 2004). For example, one could determine the conditional distributions of each μ_j parameter, which would result in 10 blocks each consisting of a two-dimensional space. To obtain samples from the joint posterior in this regime, one would then cycle through each conditional distribution and sample from this distribution using ABCDE. Because the ABCDE algorithm employs a system of Markov chains, developing such a sampler is straightforward, unlike other algorithms that rely on sequential Monte Carlo techniques. However, to create a modeling challenge for ABCDE, we instead used only a single block (a twenty-dimensional space). In this regime, rejection rates will be very high and efficiently sampling from the joint posterior distribution will be difficult.

The middle panel of Fig. 6 shows the estimated joint posterior distribution for each of the 10 bivariate normal components during the burn-in period. To illustrate the convergence of the algorithm we color-coded the parameter estimates based on the current iteration. Beginning with the uniform distribution, the particles representing the 20 parameters quickly converged to cluster around the 10 bivariate means. This is made possible because δ starts high to provide a non-zero weight for the DE crossover algorithm to converge to the high-density regions. The left panel of Fig. 6 shows the convergence of δ across the iterations and is color coded to match the posterior estimates in the middle panel. The values of δ decrease with each iteration as ABCDE converges to the high-density regions of the posterior. The right panel of Fig. 6 shows the final estimate of the joint posterior distribution of the 10 bivariate normal components obtained during the 300 sampling mode iterations following the burn-in period.

This example has illustrated a number of important features of the ABCDE algorithm. First, while the algorithm was able to converge to the bivariate means, the algorithm was able to recover the true posterior distribution and did not simply converge to single points (i.e., the maximum a posteriori estimate). Furthermore, this accuracy was obtained using only 25,000 model simulations with a total computation time of under one minute. We speculate that obtaining such results would be nearly impossible when using a standard ABC algorithm such as ABC SMC due to the extremely high rejection rates. We also implemented a KABC algorithm using the same number of particles and iterations, and a normal transition kernel with standard deviation equal to 0.1. Although we do not report these results here, the algorithm failed to converge within 500 iterations, leaving the final estimated posterior spanning the entire space. This additional simulation suggests that the crossover portion of the algorithm is essential for efficiently scaling ABC to models with large numbers of parameters.

4. General discussion

Typically, Monte Carlo methods such as Markov chain Monte Carlo or sequential Monte Carlo rely on transition kernels to move proposals around the region of interest. These transition kernels generally are Gaussian in form and are centered at the previous state of the chain. Thus, to fully specify the kernel, one only needs to select the standard deviation of the Gaussian as a tuning parameter. However, it can be very difficult to specify the optimal transition kernel, especially in the case of highly correlated parameters. For highly correlated parameter spaces (e.g., correlations greater than 0.4), Turner et al. (submitted for publication) have shown in a simulation study that the DE proposal generation method is much more

Please cite this article in press as: Turner, B. M., & Sederberg, P. B. Approximate Bayesian computation with differential evolution. Journal of Mathematical Psychology (2012), doi:10.1016/j.jmp.2012.06.004

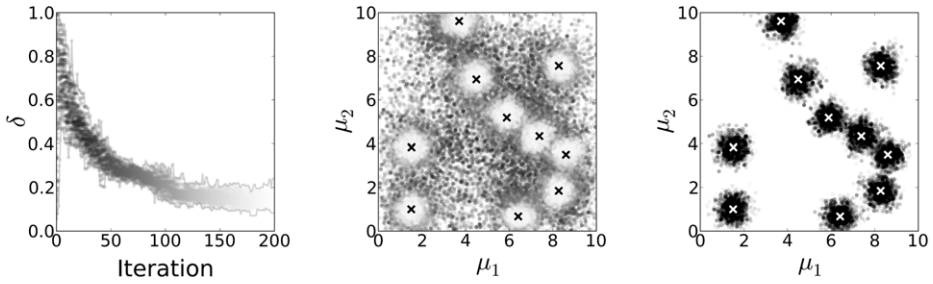

Fig. 6. The marginal convergence of all δ values (left panel) and the estimated joint posterior distribution of (μ_1, μ_2) (middle panel) as a function of the iteration number during the burn-in period. The grayscale color coding is equivalent for the left and middle panels to show the trade-off between the estimated posterior and the maximum and minimum values of the error term. The right panel plots the final estimated joint posterior distribution using only the samples obtained during the sampling period.

efficient than conventional Markov chain Monte Carlo algorithms, as measured by both the rejection rate and the Kullback–Leibler distance. Furthermore, in the case of high-dimensional parameter spaces, the cost of selecting sub-optimal transition kernels can be incredibly high—many more samples may be required to adequately sample from the posterior.

On the other hand, genetic algorithms such as DE relieve the demands of selecting transition kernels because the perturbation method is explicitly dependent on the current population of particles. In addition to DE being convenient, it has been shown that the basic DE algorithm can achieve the optimal acceptance rate when

$$\gamma_1 = \frac{2.38}{\sqrt{2d}},$$

where d is the number of dimensions in the parameter space (ter Braak, 2006). The automatic specification of a transition kernel used by ABCDE offers a compelling advantage over most current algorithms in the ABC framework, where rejection rates are exacerbated by the mandatory comparison of the distance between simulated and observed data to some tolerance threshold.

The ABCDE algorithm has another benefit that was not explored in this article. Because the ABCDE algorithm consists of a system of Markov chains, sets of parameters can be "blocked" and estimated cyclically. For example, when estimating the parameters of a hierarchical model, likelihood-informed methods can be used for the hyperparameters whose conditional distributions do not depend on the likelihood, and ABCDE can be used for the lower-level parameters as shown in Turner and Van Zandt (submitted for publication). This is an added benefit of ABCDE over sequential Monte Carlo methods, which do not converge to the true joint posterior distribution when used in a Gibbs sampling framework unless proper precautions are taken.

While the ABCDE algorithm is technically a system of Markov chains, it can still be easily parallelized by distributing model simulations in the "for" loop (Line 6 in Fig. 1). We simulated all examples presented in this article in R on a standard desktop computer with an Intel i7 processor (3.07 GHz) and distributed the jobs across eight cores. We found that this was sufficient for our needs, but we emphasize that more powerful computational tools (e.g., compiler languages, graphics processor units, computing clusters) could be used to amplify the algorithm's efficiency well beyond what we demonstrate in the present examples.

4.1. Guidelines

A difficulty of employing ABCDE is that, in its complete form with crossover, migration, and mutation, it requires a number of tuning parameters. While we have found that the selection of these parameters does not critically alter the performance of the algorithm, we now provide some basic guidelines for their selection. First, the number of particles per group should be about 5–10 times the number of modes in the posterior and/or parameters in the model. Dividing up the pool of particles is really just a matter of selecting the appropriate group size. We have found that a group size of 20–50 particles works well to fit a wide range of models.

Selecting the migration probability α and the mutation probability β is usually a matter of knowing something about the posterior distribution. While these features were not used in this article, we have found that α and β should usually be some small value (e.g., $\alpha = \beta = 0.1$). For multi-modal posteriors, it might be important to increase the α parameter to allow for a more thorough exploration of the parameter space. Selecting β and κ is a matter of convergence speed. We have found that small values such as $\beta = 0.1$ or $\beta = 0.2$ work quite well. Increasing β will result in more mutation steps, and the algorithm will be very similar to the ABC SMC algorithm (Toni et al., 2009) or the Markov chain Monte Carlo algorithm (Wilkinson, submitted for publication). Large values of the crossover parameter update probability (e.g., $\kappa = 0.9$) will allow for fast convergence while maintaining some additional diversity in the crossover proposals, which is ideal for most models and especially when there is parameter dependence. Some problems, however, may benefit from slower convergence with $\kappa = 0.2$ that fosters search along individual dimensions. Selecting the perturbation kernel in the mutation step is similar to selecting the kernel in the ABC SMC algorithm. However, for ABCDE, we do not need these transition kernels to be highly variable and have found that choosing a normal kernel with small variance works well.

Finally, the choice of the model error function $\psi(\cdot|\delta)$ is an important one. While we have found the Gaussian and Epanechnikov functions very useful, we also suggest using exponential functions. As $\delta \to 0$, we will obtain estimates closer to the true posterior distribution, and the importance of selecting the model error function $\psi(\cdot|\delta)$ will diminish. In the examples presented here, we have shown that δ can be treated as a free parameter in the model, which automates the fitting procedure.

It is important to note that while including crossover, migration, and mutation provides for the most flexible algorithm, in many cases ABCDE works extremely well with only a single group of particles that update via the crossover step. This is because the DE algorithm in combination with the Metropolis–Hastings acceptance probability allows for efficient proposals that span the entire range of high-density regions of the parameter space. The bivariate normal simulation outlined above is an example of estimating a complex model with only the crossover component. In such cases, you only need to decide on the number of particles and an error function, making the simulation quite straightforward.

5. Conclusions

In this article, we have presented a new algorithm, ABCDE, that attempts to solve the problem of scalability associated with ABC

Please cite this article in press as: Turner, B. M., & Sederberg, P. B. Approximate Bayesian computation with differential evolution. Journal of Mathematical Psychology (2012), doi:10.1016/j.jmp.2012.06.004

methods. The algorithm relies on DE to drive the sampler to high-density regions of the posterior distribution and automatically optimize the proposal efficiency. This type of proposal generation is very efficient, capturing both multi-modal and high-dimensional posteriors both quickly and accurately. Although we have not provided proofs of convergence for the ABCDE algorithm in this article, we have demonstrated with many examples (only three of which are reported here) that the ABCDE algorithm will converge to the true posterior distribution.

When developing a model in any field, one often investigates many variants of a single base model. ABC algorithms, which require only a simulation of the model, afford us the opportunity to test many variants without having to develop full statistical models for each variant. The posteriors that ABC provides can give us clear information about the relationships between the parameters—information that might otherwise be available only through extensive experience with each variant. Equipped with only a prior distribution and the ABCDE algorithm, estimation is now feasible for virtually any posterior distribution.

References

Barthelmé, S., & Chopin, N. (2011). ABC-EP: expectation propagation for likelihood-free Bayesian computation. In *Proceedings of the 28th international conference on machine learning*. Bellevue, WA.

Beaumont, M. A. (2010). Approximate Bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution, and Systematics*, *41*, 379–406.

Beaumont, M. A., Cornuet, J.-M., Marin, J.-M., & Robert, C. P. (2009). Adaptive approximate Bayesian computation. *Biometrika*, *asp052*, 1–8.

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

Blum, M. G. B., & François, O. (2010). Non-linear regression models for approximate Bayesian computation. *Statistics and Computing*, *20*, 63–73.

Chhikara, R. S., & Folks, L. (1989). *The inverse Gaussian distribution: theory methodology and applications*. New York: Marcel Dekker, Inc.

Craigmile, P., Peruggia, M., & Van Zandt, T. (2010). Hierarchical Bayes models for response time data. *Psychometrika*, *75*, 613–632.

Del Moral, P., Doucet, A., & Jasra, A. (2008). An adaptive sequential Monte Carlo method for approximate Bayesian computation Technical Report.

Feller, W. (1968). *An introduction to probability theory and its applications, Vol. 1*. New York: John Wiley.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis*. New York (NY): Chapman and Hall.

Havangi, R., Nekoui, M., & Teshnehlab, M. (2010). A multi swarm particle filter for mobile robot localization. *International Journal of Computer Science*, (7), 15–22.

Heathcote, A. (2004). Fitting Wald and ex-Wald distributions to response time data: an example using functions for the S-PLUS package. *Behavioral Research Methods, Instruments, & Computers*, *36*, 678–694.

Hu, B., & Tsui, K.-W. (2005). Distributed evolutionary Monte Carlo with applications to Bayesian analysis Technical Report Number 1112.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks. vol. 4* (pp. 1942–1948).

Lee, M. D., Fuss, I. G., & Navarro, D. J. (2006). A Bayesian approach to diffusion models of decision-making and response time. In B. Scholkopf, J. Platt, & T. Hoffman (Eds.), *Advances in neural information processing* (19th ed.) (pp. 809–815). Cambridge, MA: MIT Press.

Lee, M.D., & Wagenmakers, E.-J. (2012). A course in Bayesian graphical modeling for cognitive science, last downloaded January 1, 2012. Available from http://www.ejwagenmakers.com/BayesCourse/BayesBookWeb.pdf.

Luce, R. D. (1986). *Response times: their role in inferring elementary mental organization*. New York: Oxford University Press.

Marjoram, P., Molitor, J., Plagnol, V., & Tavare, S. (2003). Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the United States*, *100*, 324–328.

Matzke, D., & Wagenmakers, E.-J. (2009). Psychological interpretation of the ex-Gaussian and shifted wald parameters: a diffusion model analysis. *Psychonomic Bulletin and Review*, *16*, 798–817.

Navarro, D. J., & Fuss, I. G. (2009). Fast and accurate calculations for first-passage times in Wiener diffusion models. *Journal of Mathematical Psychology*, *53*, 222–230.

Oravecz, Z., Tuerlinckx, F., & Vandekerckhove, J. (2009). A hierarchical Ornstein–Uhlenbeck model for continuous repeated measurement data. *Psychometrika*, *74*, 395–418.

Peruggia, M., Van Zandt, T., & Chen, M. (2002). Was it a car or a cat i saw? An analysis of response times for word recognition. *Case Studies in Bayesian Statistics*, *VI*, 319–334.

Pritchard, J. K., Seielstad, M. T., Perez-Lezaun, A., & Feldman, M. W. (1999). Population growth of human *Y* chromosomes: a study of *Y* chromosome microsatellites. *Molecular Biology and Evolution*, *16*, 1791–1798.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, *85*, 59–108.

Rice, J. A. (2007). *Mathematical statistics and data analysis*. Belmont, CA: Duxbury Press.

Robert, C. P., Cornuet, J.-M., Marin, J.-M., & Pillai, N. (2011). Lack of confidence in approximate Bayesian computation model choice. *Proceedings of the National Academy of Sciences of the United States*, *108*, 15112–15117.

Rouder, J. N., Lu, J., Speckman, P., Sun, D., & Jiang, Y. (2005). A hierarchical model for estimating response time distributions. *Psychonomic Bulletin and Review*, *12*, 195–223.

Rouder, J., Sun, D., Speckman, P., Lu, J., & Zhou, D. (2003). A hierarchical Bayesian statistical framework for response time distributions. *Psychometrika*, *68*, 589–606.

Schwarz, W. (2001). The ex-Wald distribution as a descriptive model of response times. *Behavioral Research Methods, Instruments, & Computers*, *33*, 457–469.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. London: Chapman & Hall.

Sisson, S., Fan, Y., & Tanaka, M. M. (2007). Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the United States*, *104*, 1760–1765.

Storn, R., & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*, 341–359.

Tanese, R. (1989). *Distributed genetic algorithms* (pp. 434–439).

Tavaré, S., Balding, D. J., Griffiths, R. C., & Donnelly, P. (1997). Inferring coalescence times from dna sequence data. *Genetics*, *145*, 505–518.

ter Braak, C. J. F. (2006). A Markov chain Monte Carlo version of the genetic algorithm differential evolution: easy Bayesian computing for real parameter spaces. *Statistics and Computing*, *16*, 239–249.

Tong, G., Fang, Z., & Xu, X. (2006). A particle swarm optimized particle filter for nonlinear system state estimation. *Evolutionary Computation*, 438–442.

Toni, T., Welch, D., Strelkowa, N., Ipsen, A., & Stumpf, M. P. (2009). Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, *6*, 187–202.

Turner, B.M., Sederberg, P.B., Brown, S.D., & Steyvers, M. (2012). A note on efficiently sampling from distributions with correlated dimensions, manuscript (submitted for publication).

Turner, B.M., & Van Zandt, T. (2012). Hierarchical approximate Bayesian computation, manuscript (submitted for publication).

Turner, B. M., & Van Zandt, T. (2012). A tutorial on approximate Bayesian computation. *Journal of Mathematical Psychology*, *56*, 69–85.

Vandekerckhove, J., Tuerlinckx, F., & Lee, M. D. (2011). Hierarchical diffusion models for two-choice response time. *Psychological Methods*, *16*, 44–62.

Vrugt, J. A., ter Braak, C. J. F., Diks, C. G. H., Robinson, B. A., Hyman, J. M., & Higdon, D. (2009). Accelerating Markov chain Monte Carlo simulation by differential evolution with self-adaptive randomized subspace sampling. *International Journal of Nonlinear Sciences and Numerical Simulation*, 10.

Wald, A. (1947). *Sequential analysis*. New York: Wiley.

Wilkinson, R.D. (2011). Approximate Bayesian computation (ABC) gives exact results under the assumption of model error, manuscript (submitted for publication).

Please cite this article in press as: Turner, B. M., & Sederberg, P. B. Approximate Bayesian computation with differential evolution. Journal of Mathematical Psychology (2012), doi:10.1016/j.jmp.2012.06.004