Parallel Probability Density Approximation

Yi-Shin Lin and Andrew Heathcote

University of Tasmania

William R. Holmes

Vanderbilt University

Author Note

Yi-Shin Lin and Andrew Heathcote, Division of Psychology, School of Medicine, University of Tasmania, Australia.

William R. Holmes, Department of Physics and Astronomy & Department of Mathematics, Quantitative Systems Biology Center, Vanderbilt University, USA. WRH was supported by National Science Foundation (USA) Grant SES-1530760. AH is supported by Australian Research Council Discovery Project DP160101891

Correspondence concerning this article should be addressed to Yi-Shin Lin, Division of Psychology, School of Medicine, University of Tasmania, Private Bag 30, Hobart, TAS, 7005, Australia. Contact: yishin.lin@utas.edu.au

Abstract

Probability Density Approximation (PDA) is a non-parametric method of calculating probability densities. When integrated into Bayesian estimation, it allows researchers to fit psychological processes for which analytic probability functions are unavailable, significantly expanding the scope of theories that can be quantitatively tested. PDA is, however, computationally intensive, requiring large numbers of Monte Carlo simulations to attain good precision. We introduce Parallel PDA (pPDA), a highly efficient implementation of this method utilizing Armadillo C++ and CUDA C libraries to conduct millions of model simulations simultaneously in graphics processing units (GPUs). This approach provides a practical solution for rapidly approximating probability densities with high precision. In addition to demonstrating this method, we fit a Piecewise Linear Ballistic Accumulator model (Holmes, Trueblood & Heathcote, 2016) to empirical data. Finally, we conduct simulation studies to investigate various issues associated with the PDA and provide guidelines for pPDA applications to other complex cognitive models.

Keywords: R, C++, CUDA, GPU, kernel density estimate, Markov Chain Monte Carlo, Bayesian modeling, Probability Density Approximation

Parallel Probability Density Approximation

With the rapidly increasing capabilities of computer hardware and software in recent years, simulation-based approaches to investigate mathematical models of various phenomena have exploded. In the long history of using cognitive models to test theories some of work has been qualitative in nature, seeing whether a theory can predict observed patterns of data. Other work has been quantitative in nature, determining whether a model / theory can quantitively match the features in data. The latter approach typically involves fitting a model to data to determine how well the theory encoded in that model match observations. Most (though not all) past methods of achieving this have typically been limited to relatively simple models which have tractable likelihood functions. More recently however, new simulation based methods have been developed (Palestro, Sederberg, Osth, van Zandt, & Turner, 2018) that utilize modern computational power to significantly expand the scope of models that quantitative fitting can be applied to. Here, we describe a highly efficient, GPU enabled parallel implementation of canonical Bayesian Markov chain Monte Carlo (MCMC) methods utilizing the *Probability Density Approximation* (*PDA*) (Holmes, 2015; Turner & Sederberg, 2014).

MCMC methods can be used to preform either Bayesian Computation or Approximate Bayesian Computation (ABC) (Beaumont, 2010; Sisson & Fan, 2010). The former requires the likelihood for the model of interest to be analytically tractable (i.e., either solvable in terms of fundamental functions or amenable to fast, stable and accurate approximation through standard numerical integration methods) while the latter does not. The likelihood refers to the probability of observing a given data set for a given vector of model parameters. As hypotheses become more fine-grained and realistic, so do the resulting models. These more descriptive models, however, typically also become more complex and have analytically intractable likelihoods.

ABC has helped overcome the limitations imposed this intractability by utilizing large scale computation.

At its heart, the PDA method can be utilized in any standard Bayesian MCMC framework by directly replacing an analytically calculated likelihood function with a numerically approximated one derived from large numbers of model simulations. This approach overcomes several numerical and statistical issues that are often found with ABC methods. To assess the quality of fit of a model / parameter combination, typical ABC methods simulate from the model a large synthetic data set and compare it to empirical data. This comparison often relies on a set of summary statistics (e.g., the mean and variance), where those statistics are calculated for both data and model simulations. This raises two central issues that are addressed by the PDA. First, if those statistics are not "sufficient" (i.e., capturing all available information), this process compresses both the model and data, potentially introducing errors. Second, a likelihood, which is the foundation of Bayesian methods, is never calculated. The PDA circumvents both issues by applying non-parametric Kernel Density Estimation [KDE, (Silverman, 1986)] to the simulated data to numerically approximate the model's likelihood. The resulting summary is both sufficient and can be directly plugged into any procedure requiring likelihoods, such as Bayesian MCMC methods.

The PDA method was introduced to psychology by Turner and Sederberg (2014). A more efficient variant was derived subsequently by Holmes (2015), using Silverman's (1982) KDE algorithm. In short, Silverman uses a Fast Fourier Transform (FFT) to reduce the computational burden of the KDE by utilizing a Gaussian kernel function and transforming calculation of the KDE to the spectral domain. Holmes's implementation also utilizes likelihood resampling to reduce MCMC chain stagnation arising from likelihood approximation errors (see Holmes, for

further discussion). In this paper, we address both the issue of computational burden of this method through an efficient parallel implementation as well as explore the influence of likelihood resampling on the mixing of Markov chains.

The PDA methodology is attractive because it bypasses the need to construct formal probability density functions, providing a tractable path for psychologists to test their conceptual theories by converting them into simulation models. The essential benefit of this approach is that it is mathematically and conceptually much simpler; simulating models directly is more straightforward and requires less person time than mathematically deriving complicated likelihood functions. Although several popular psychological models do have tractable likelihood functions, such as the exponentially modified Gaussian model (Luce, 1986), and the diffusion decision model (Ratcliff & McKoon, 2008), many do not (e.g., Thura, Beauregard-Racine, Fradet, Cisek, 2012; Gureckis, & Love, 2009; Tsetsos, Usher, & McClelland, 2011; Cisek, Puskas, & El-Murr, 2009). In many cases, even those that are tractable can require an enormous investment of skilled human time. For example, the DDM, one of the most popular response time modeling frameworks, has seen significant sustained research aimed at more efficiently and accurately calculating the model likelihood. At best, this limitation significantly slows scientific progress due to the effort required to mathematically derive the complex formulas for these likelihoods. At worst, it may restrict the range of questions to which modeling is applied. In fact, one could argue that many of the most popular and widely used cognitive models have become so in part because they are accessible. The goal of the PDA, and simulation based methods more broadly, is two-fold. First, to expand the scope theories / models that can be tested. Second, to trade human time (less time deriving mathematical formulae) for computer time (more time simulating models).

Although the PDA method does free up researcher time to address psychological questions rather than mathematical technicalities, it does present its own set of issues, computational time being one of the most significant. It suffers from two computational bottlenecks. The first is calculating the KDE, which in the general case requires a discrete convolution. The second is generating enough model simulations to obtain a sufficiently accurate approximation. These computational bottlenecks are aggravated when applying PDA in Bayesian modeling context, since these operations must be performed iteratively for multiple Markov chains. Furthermore, these operations introduce approximation errors which need to be minimized (Holmes, 2015; Turner & Sederberg, 2014). Hence an efficient computational method is critical not just for improving performance, but also for minimizing errors.

Our solution, Parallel PDA (pPDA), is coded in Armadillo C++, a highly efficient C++ library for linear algebra (Sanderson & Curtin, 2016) and Compute Unified Device Architecture, CUDA C, a programming model accessing the power of graphics processing units (GPUs). This heterogeneous programming model allows pPDA to harness the strengthens of CPU and GPU to attain efficient and precise likelihood estimates. In a nutshell, the GPU conducts numerous (e.g., millions) model simulations, which are then used to construct a histogram that is passed back to the CPU for further analysis. This part is coded in CUDA C. Next, the CPU applies Silverman's (1982) KDE algorithm to approximate the log likelihood of each individual data observation. The latter part is coded in Armadillo C++. This implementation allocates the heavy burden of simulating many independent model samples to the GPU, with the resulting summarized data transferred back to the CPU for subsequent processing. In between these two steps, we optimize what is transferred to and from the GPU to minimize the data transfer burden that often plagues GPU based computations. To ease the installation and usage of pPDA, we created an open

source R (R Core Team, 2017) package, *ppda*, made available at our GitHub (*https://github.com/TasCL/ppda*). The user can easily access *ppda* using regular R installation methods.

In this paper, we first give a brief overview of the PDA method implemented in *ppda* and the difficulties associated with it. Because PDA, MCMC, and GPU-based parallel computation are relatively new techniques, we first illustrate the application of PDA by itself to six well-known cognitive models before integrating the three techniques together. Next, we present a series of model-base studies to investigate the validity, the scope, and the best practice of all three techniques combined in the *ppda* package. These studies demonstrate the ability of pPDA to solve a cutting-edge problem in cognitive modeling, fitting the Piecewise Linear Ballistic Accumulator (PLBA) model (Holmes, Trueblood, & Heathcote, 2016), which does not have an easily computed likelihood. We conclude the paper with a discussion about the specific issue of inflated likelihood in the PDA, and roadmaps for applying pPDA to other cognitive models and the future development of the *ppda* package. We have provided a detailed account of all analyses reported in this paper at https://osf.io/p4pdh as a model for users who wish to make their own applications.

Probability Density Approximation

In this section, we give an overview of the PDA and its application in Bayesian computation (see Turner & Sederberg, 2014, and Holmes, 2015, for further details, and general reviews of ABC in Sisson & Fan, 2010 and Beaumont, 2010).

Bayesian inference derives a posterior distribution for a set of parameters (θ) given the data (y) by multiplying a prior distribution, $\pi(\theta)$ by the model likelihood, $\pi(y|\theta)$ according to Bayes' rule:

$$\pi(\theta|y) = \frac{\pi(y|\theta)\pi(\theta)}{\int \pi(y|\theta)\pi(\theta)d\theta} \qquad (1)$$

Bayes estimation can be carried out by MCMC methods without evaluating the denominator in the equation (1) as the integral is often intractable using:

$$\pi(\theta|y) \propto \pi(y|\theta)\pi(\theta) \qquad (2)$$

Metropolis-based MCMC methods operate iteratively by proposing new sets of parameters, denoted θ^*. There are several ways to propose parameters, such as from a multivariate Gaussian distribution (Gelman, 2014), or in other adaptive ways that optimize the proposals (e.g., Roberts & Rosenthal, 2001; Hoffman & Gelman, 2014; Ter Braak, 2006; Turner & Sederberg, 2012; Neal, 1994). Together with a data set, one can derive a number proportional to the posterior probability density by calculating the right-hand side of equation (2). This proposal probability density is then compared to the probability density from a previous iteration (a "reference" density). When using a symmetric jumping distribution (e.g., a multivariate Gaussian distribution), this comparison usually involves a ratio of the proposal density to the reference density, so the denominator in equation (1) (which is not a function of the current parameters) is irrelevant as it cancels out. The ratio, $\frac{\pi(\theta^*|y)}{\pi(\theta^{i-1}|y)}$, then guides an accept vs. reject step deciding whether the proposed parameters are more probable than the reference parameters; if they are, they are accepted (i.e., they replace the reference parameters), and if not, they may still be accepted with a probability proportional to the ratio, and are otherwise rejected. The acceptance for less probable proposals sampling from low density regions, which is necessary because the aim of Bayesian computation is to recover the full target posterior distribution. The reference parameters are then kept for the next iteration. The critical element of this process is the

computation of the likelihood of existing and proposed values of model parameters to be compared. The lack of an analytic function to calculate these values is precisely the issue that PDA (and essentially all ABC methods for that matter) helps overcome.

At the most basic level, PDA is a technique that facilitates the application of MCMC methods to analytically intractable models, rather than a completely new method itself. More precisely, it can be integrated into any MCMC based procedure that relies on the likelihood function to calculate parameter acceptance rates. Thus, it could in principle be integrated into any Metropolis-Hastings MCMC framework. Given this generality, we do not describe here a particular MCMC algorithm, but note that in the applications that follow and associated software, we will use the Differential Evolution MCMC (DE-MCMC, Turner, Sederberg, Brown, & Steyvers, 2013) method. For further description of MCMC methods, see Brooks, Gelman, Jones, & Meng (2011).

PDA can be integrated into any MCMC procedure by replacing the likelihood $\pi(y|\theta)$ with a numerically approximated likelihood function $\tilde{\pi}(y|\, x, \theta)$, where *x* denotes a large data synthetic data set simulated from the model with parameters θ. From here on $\tilde{}$ will indicate an approximate quantity. This renders the posterior probability density function:

$$\pi(\theta|y) \propto \tilde{\pi}(y|x,\theta)\pi(\theta) \qquad (3)$$

Where this relation is now an approximate rather than exact proportionality due the approximation of the likelihood.

The key now is to generate the approximation $\tilde{\pi}(y|x,\theta)$. To accomplish this, three steps are required. The first is to simulate a large number of responses from the model. This is of course model dependent, but is typically a simple, though computationally intensive process. In the case of a response time model, this would require simulating many choice / response time

pairs. The second step, is to construct an approximate log likelihood function on a discrete grid (a grid of time points for example) from those simulations using Kernel Density Estimation (described below). Finally, the log likelihoods of each individual data observation are approximated by interpolating their value from the discrete, grid based approximation just constructed.

We now describe the kernel density estimation process. The kernel density estimate at a point y_i is

$$\hat{f}(y_i) = \frac{1}{N_s}\sum_{j=1}^{N_s} K_h(y_i - x_j) \qquad (4)$$

where x_j are the simulated data points and K_h is a smoothing kernel that satisfies

$$K_h(z) = \frac{1}{h}K(\frac{z}{h}) \qquad (5)$$

Where *K* integrates to one over the domain of interest. The kernel function here could be any standard kernel. The specific mathematical trick used here to improve performance here however requires the use of a Gaussian kernel. The KDE-FFT method improves computational efficiency by recognizing that the KDE can be reformulated as a convolution of the data against the smoothing kernel

$$\hat{f}(y_i) = \tilde{d} \star K_h(z) \qquad (6)$$

where $\tilde{d}$ is a histogram of the simulated data, resulting from binning the data set *x* over a very fine (but hence noisy) grid of equal intervals. Convolutions become a multiplication operation (which is much more efficient) when transformed into the frequency domain. Thus, the KDE-FFT computes the density by first transforming both the simulated histogram and the kernel function to the frequency domain, conducting multiplication and transforming the result back. Hence, an efficient way to derive the approximated probability densities is:

$$\hat{f}(y) = \mathcal{F}^{-1}\left(\mathcal{F}\left[\tilde{d}\right] \cdot \mathcal{F}[K_h]\right) \qquad (7)$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are respectively FFT and inverse FFT operations. Here, we use a Gaussian kernel, which has the advantage that the Fourier transform of Gaussian is another Gaussian. That is, given that K_H is a Gaussian, $\mathcal{F}[\mathrm{K_h}]$ is also a Gaussian with well known form.

The detailed steps to computing the approximation of the likelihood of the data are as follows. 1) Simulate a large data set *x*. 2) Bin that data set into a very fine but noisy histogram $\tilde{\mathrm{d}}$ on a regular grid. 3) Fourier transform that discrete data function to produce $\mathcal{F}\left[\tilde{\mathrm{d}}\right]$. 4) Multiply that transformed function by the transformed kernel $\mathcal{F}[\mathrm{K_h}]$. 5) Calculate the inverse transform. These first five steps produce a discrete approximation of the likelihood function. 6) Use linear interpolation to approximate the likelihood values at the data points in the set *y*. We note that the only step in this process that is dependent on the specific model under consideration is (1), the model simulation.

Cognitive Models

We illustrate probability density approximation in six well-known cognitive models to demonstrate, in a simple and easily explained context, that PDA is a general method, before incorporating it into approximation Bayesian computation. These six models are ex-Gaussian, gamma, Wald, and Weibull distributions, diffusion decision (Ratcliff & McKoon, 2008) and linear ballistic accumulation (Brown & Heathcote, 2008) models. We choose these six oft-used models in cognitive literature because their likelihoods are relatively easy to compute, have been thoroughly tested (Ratcliff, 1978; Van Zandt, 2000; Turner & Sederberg, 2014), and are useful in fitting response time data. Because all six models have formal likelihood equations, this illustration shows that PDA works for a wide range of models, that it can replicate previous work, and that it is relevant to cognitive modeling.

The first example, the ex-Gaussian distribution, is a convolution (i.e., sum) of exponential and Gaussian random variables. The ex-Gaussian was originally proposed as a two-stage cognitive processes, with a Gaussian component associated with perceptual and response-production time and an exponential component associated with decision time (Hohle, 1965; Dawson, 1988). Although now considered only a descriptive model (Matzke & Wagenmakers, 2009), the ex-Gaussian is a good approximation to the shape of response time (RT) distributions of the data collected in typical RT experiments. The second example is the gamma distribution, which is traditionally used to model cognitive serial processes, because it is the convolution of a series of exponential steps, and the exponential component is often associated with decision processes (McClelland, 1979).

[Insert Figure 1 Here]

Figure 1a shows the result of approximating the ex-Gaussian model. We used three different methods to calculate ex-Gaussian probability density functions (PDFs). The first method derives approximated densities directly from a Gaussian kernel, represented by a gray dotted line. This is the traditional kernel density estimation (KDE) (Van Zandt, 2000). The second method uses KDE-FFT method to estimate densities (Silverman, 1982), represented by a gray dashed line, and the third directly calculates the ex-Gaussian PDF, represented by a dark dotted line. Both KDE methods matched the analytic solution with good precision. This demonstrates two points. First, accurate approximations of common densities can be readily constructed. Second, comparing to the traditional KDE, the KDE-FFT introduces essentially no error. Therefore, from here on, we compare only the KDE-FFT to the analytic solution.

The next three examples are the gamma, the Wald and Weibull models, shown in Figure 1b, 1c and 1d. The Wald model describes continuous one-boundary diffusion process and can be

used to account for RT data from simple detection experiments where observers make only one type of response (Schwarz, 2002; Heathcote, 2004). The Weibull model is used to describe RT data resulting from the asymptotic distribution of the minimum completion time of a large number parallel racing processes. Again, PDA estimation matches analytic solutions in these three models.

The next two examples, the diffusion-decision model (Ratcliff, 1978) and the LBA model (Brown & Heathcote, 2008), are process models. These two examples show PDA can be applied to cognitive models, accounting for RT data from binary choice tasks. For example, in a binary choice visual search task, observers might decide whether a target is found on the right or left side of the visual field. In this example, a data point consists of a pair of numbers: how much time observer takes to respond (i.e., RT) and which choice is made (i.e., right or the left). Although we use binary search as an example, the methods discussed are not limited to this context and could in principle be easily extended to models of more than two choices. Each choice is associated with a response time distribution describing the likelihood of that choice being made at a given time. Each distribution can be defective in the sense that it integrates to the probability of making that particular choice, which can be less than one.

The LBA accounts for choice as a race between independent, deterministic accumulators, where each accumulator represents a decision variable associated with the amount of evidence for a particular choice (e.g., N choices would be associated with N accumulators). A choice is made when one of those accumulators crosses a threshold representing the level of "caution" or the amount of evidence required to reach a decision. RT is the sum of the time to make a choice and a non-decision time (constituted of the time to encode a stimulus and the time to produce a response). There are three main types of parameters associated with this modeling framework:

accumulator start point, rate of accumulation, and response threshold. In the canonical LBA, start points are assumed to be vary from trial to trial and are taken to be uniform distributed between 0 and *A*. The rate of evidence accumulation is also assumed to vary from trial to trial, reflecting imperfect encoding of stimulus information, and is described by a normal distribution. Finally, the response threshold is assumed to be fixed from trial to trial (equivalently it could have a uniform distribution and the start point could be fixed) at a distance *B* above *A*. With this information we can simulate experimental trials to generate a synthetic data set, which can be subsequently used to produce a collection of defective probability density functions (PDF) representing the RT distributions to compare against data. A similar process can be applied to the diffusion-decision model (Figure 1f), which assumes a single accumulation process with two boundaries, one for each response, and so it is limited to modeling binary choice. Evidence varies from moment-to-moment during accumulation, with the average rate varying normally and the starting point of accumulation varying uniformly from trial-trial.

Results in Figure 1e and 1f show that using simulated data in conjunction with KDE produces RT distributions that closely matches analytic PDFs, with the only apparent differences being in the high likelihood region around the mode of the distributions. As will be discussed later, this slight mismatch in high curvature regions of the distributions are expected as errors in the KDE are related to the local curvature of the density to be estimated, which for distributions of the form tested here occurs at the mode. These two cases illustrate one important advantage of PDA: it is a general method that can easily applies to different cognitive models. Although the two models can either be calculated analytically or approximated with good precision, even slight modifications of them make them analytically intractable. PDA can easily handle modifications that encode more complex mechanisms.

Monte Carlo Bottleneck

One major issue with PDA is that it requires many model simulations to generate an accurate approximation of the likelihood function. As we will show in the simulation studies, when data are noisy, the added noise introduced by the PDA approximation will affect one's ability to recover model parameters. A solution for this problem is to increase the number of model simulations, but this incurs computational costs that quickly escalate. Consequently, the first issue often constrains researchers from gaining strong confidence in the accuracy of their approximation.

To see why model simulations may result in a heavy computation burden, consider "real-world example 2" in Turner and Sederberg (2014). They estimated likelihoods for each proposal with 30,000 model simulations for 5,000 MCMC iterations plus a 1,000 burn-in period on 36 separate Markov chains. Thus, they had to draw 6.48 trillion random numbers for each of 34 participants, a large burden, even for modern CPUs, so that even a small increase in model simulations, for example from 30,000 to 40,000, will quickly render the computation prohibitively expensive.

One possible solution, discussed in Turner and Sederberg (2014) and explored in another similar simulation-based method (Verdonck, Meers, & Tuerlinckx, 2016), is to conduct parallel Monte Carlo simulations via GPUs. Although this is promising, an effective algorithm to deliver this promise remains to be created. The development of a parallel method for PDA to overcome these computational issues is the subject of the remainder of this article.

Parallel Probability Density Approximation

To resolve this dilemma for evidence accumulation models, pPDA transports simulation parameters into GPU memory and then designates two memory pointers to the locations where

the outcomes of simulations will be stored. Two pointers are required because it fits data with two dependent variables (i.e., choices and RTs). For the models with one dependent variable, such as the ex-Gaussian, only one memory pointer is needed. The information sent into the GPU memory consists of only the number of model simulation and the model parameters, typically only a few unsigned integers and floating-point numbers, which is well below the bandwidth limit of most GPUs. After the simulations are generated and stored temporarily inside the GPU memory, pPDA deploys five CUDA functions[1], operating in GPU memory to extract information required for kernel density estimation back to CPU memory. These functions calculate the numbers of each choice, the maximum, and the minimum of simulated RTs, and their standard deviation[2]. These statistics are then used to construct kernel bandwidths, the ranges of the bin edges, bin edges, and the Gaussian filter. These operations are conducted in CPU memory. Next, pPDA transports the bin edge vector back to GPU memory to construct a histogram, which is then transported back to CPU memory. The histogram carries only 1024 unsigned integers, so again pPDA operates well below the limitation of GPU bandwidth. More details about the pPDA implementation can be found on GitHub.

Replication Study

Because pPDA is a novel method, we first examine whether it replicates results from a previous PDA study (Turner & Sederberg, 2014).

---

[1] The programming functions written in CUDA C and operating inside the GPU are dubbed kernel functions, which are not the kernel function we refer to in equation (4). Also, the path width that can accommodate amount of information exchange effectively between CPUs and GPUs is dubbed bandwidth, which is not the same as the kernel bandwidth in equation (4)

[2] We use the formula, $\sqrt{\frac{\sum x^2-(\sum x)^2/n}{n-1}}$, to calculate standard deviation, so the fourth and fifth CUDA functions extract the sum and squared sum.

Method

We deployed fifteen Markov chains, using the Differential Evolution Markov chain Monte Carlo (DE-MCMC, Turner, Sederberg, Brown, & Steyvers, 2013) sampler, which is an adaptation of the two genetic operators, *crossover* (Ter Braak, 2006) and *migration* (Hu & Tsui, 2005). Each chain drew 5,000 samples, following 8,000 burn-in samples. During the burn-in, we probabilistically used the *migration* operator (Turner et al., 2013, p383-384) by drawing a random number from a uniform distribution in every iteration. When the number was less than 0.05, the *migration* operator replaced the *crossover* operator to propose parameters. Even with such a long burn-in, some cases with small numbers of model simulations and large bandwidths had not yet reached converged, reaffirming the crucial roles of approximation precision and adequate bandwidths. Note the long burn-in is far more than necessary when using the analytic likelihood (see Turner & Sederberg, 2014). The long burn-in and generous size of the final set of posterior samples (75,000) lends us confidence that, if parameter recovery studies fail, this is likely the result of insufficient model simulations or inadequate bandwidths.

To get a sense of bias and variability, we conducted one hundred independent fits for each case, where each time a new data set was generated from the LBA model. We conducted Bayesian modeling via a collection of R functions, dubbed Dynamic Models of Choice (DMC), which can be downloaded at https://osf.io/5yeh4/ (for an overview see Heathcote et al., 2018).

Similar to Turner and Sederberg (2014), we simulated responses from the LBA model and set the threshold parameter, $b\ (= B + A)$, to 1.0, the upper bound of start point, A, to 0.75, the drift rate for correct responses, μ_{v_c} to 2.5, the drift rate for the error responses, μ_{v_e}, to 1.5, and the non-decision time, t_0, to 0.2. Following their parameter set-up, we also set the standard deviation of the drift rate to 1 as a scaling (constant) parameter for the LBA model. Then, we

conducted three parameter recovery studies. We used the analytic likelihood function (Brown & Heathcote, 2008) to estimate a data set with 10,000 trials per condition, serving as a reference with minimal data variability. Then we conducted 16,384 (i.e., 2^{14}) model simulations, slightly over 1.6 times, comparing to the number used previously (Turner & Sederberg, 2014). The number of model simulations must be a power of 2 because we use two specific parallel programming techniques, parallel reduction and for-loop unrolling in CUDA code (Harris, 2007) to speed GPU computation.

Results

Figure 2 shows the results of posterior parameter distributions estimated by pPDA (green) comparing with those estimated by the analytic approaches (orange & purple). The marginal distributions for each parameter were plotted together on one canvas, using the same bin widths. It shows that, as expected in a comparison of the two analytic approaches, a larger data sample size supports more accurate and precise parameter estimation. Also, as expected given the modest number of model simulations and attendant substantial Monte Carlo variability, pPDA produces less accurate and precise parameter estimation than the analytic estimates using the same data sample size. Although showing much larger variability than their analytic counterpart, the PDA distributions cover the true parameter values, as shown in their 95% credible interval (upper panel in Figure 2) and are comparable to those reported by Turner and Sederberg (2014; see their Figure 3 upper panel).

Figure 3 shows when the sample sizes are the same (500 trials), PDA and the analytic method attain equally good fits. The goodness-of-fit figure was constructed by using posterior predictive (parameter) values to simulate new data, and by plotting the new data with the original data. Credible intervals on the model predictions for PDA are larger, reflecting the additional

uncertainty caused by Monte Carlo variability. This variability also affects the Deviance Information Criterion (DIC), a measure reflecting goodness-of-fit with a penalty added for model complexity that is based on MCMC posterior likelihoods (Spiegelhalter, Best, Carlin, & Van Der Linde, 2002). On average, DIC (-294.63) was substantially larger for pPDA than for the analytic likelihood function (-425.16). This occurs (1) because the complexity penalty equals the variance of the posterior deviance (where the deviance is twice minus the logarithm of the likelihood)[3], and (2) because Monte Carlo error increases variability of the posterior likelihood estimates, inflating DIC. This illustrates why Holmes (2015, p19) cautioned against comparing DIC for model fit by PDA with DIC for models fit using analytic likelihoods. The same issue will apply in the comparison of DIC between PDA methods that inflate posterior likelihood variability to different degrees (e.g., by using different numbers of model simulations).

[Insert Figure 2 and Figure 3 Here]

In summary, we verify pPDA is compatible with the previous PDA implementation (Turner & Sederberg, 2014), and highlights the critical role of the number of model simulations on the estimation and likelihood variability.

Fitting PLBA Model

PDA becomes essential when a cognitive model does not have a formal likelihood function. We applied pPDA to a real-world example, a random-dot-motion (RDM) discrimination task (Ball & Sekuler, 1982), where coherent motion in one direction switches to a

[3]A variety of estimates of the DIC complexity penalty are in use. The most commonly used estimate (and the one we use here), which is proportional to the difference between the mean deviance and the deviance of the mean of the sampled parameters, has the same property of being inflated by an increase in the variability of the posterior likelihoods. This occurs because it measures the distance between the middle of the posterior deviance distribution (i.e., its mean) and its leading edge (as the deviance of the mean parameters is an estimate of the minimum deviance).

different direction midway through stimulus presentation (Holmes et al., 2016). To accommodate the motion switch, standard models, such as the LBA and diffusion decision models need to be altered because they assume a constant input. It is a relatively straightforward to simulate a change of input in these models, but deriving the corresponding formal likelihood equation requires the solution of an intractable two-dimensional integral. In contrast, PDA enabled Holmes et al. to conveniently fit and test not only models with changing inputs, but also further variants with features such as a delay in when the stimulus change affects the rate of accumulation and a delayed change in response thresholds.

The PLBA model, in its simplest form, is a series of two LBA models (Brown & Heathcote, 2008), one governing the process before the switch and the other after the switch, possibly at a delay. Specifically, we fit Holmes et al.'s data with their PLBA model *1f*, which uses two different normal distributions to draw drift rates, one for the process before the switch and the other for the process after the switch has its effect. The broader model allows the change to affect the drift rate, the threshold or both, with model *1f* assuming only the drift rate is affected. Instead of fitting hierarchical PLBA model as done by Holmes et al. (2016), we fit the model to each participant's data separately. In appendix, we present a thorough examination of the PLBA model with three other simulation studies.

Methods

We fit both the PLBA *1f* and the standard LBA models to Holmes et al.'s (2016) data to see which provided the best account. We used zero truncated normal prior distributions for the threshold and the upper bound of the start point and unbounded normal distributions for the drift rates, all with a mean of three and a standard deviation of one. For *rD* and t_0, we used uniform distributions with bounds of zero and 1.

We deployed 24 chains and retained one sample every 8th iteration. During the first 100 iterations, we used *crossover* and *migration* (5%) operators and then only *crossover* operator for the rest of iterations. We kept running the iterations until all chains were well-mixed, stationary and had at least 1,024 effective samples. For both the PLBA and the LBA fits, we used 1,048,576 model simulations and recalculated likelihoods every 4^{th} iteration. To optimize the speed, we used the R package, *ggdmc* (https://github.com/yxlin/ggdmc), which implements the DE-MCMC sampler (Turner et al., 2013) in C++. Further, we set up a two-layer parallel computation to reduce computational times. Briefly, we divided 31 participants into three groups of 11, 10 and 10. Three groups of participants were fit in parallel in three identical virtual machines. Each machine is equipped with one Nvidia K80 GPU and one 12-core Intel CPU. Then we run a pseudo parallel scheme in each virtual machine, allowing multiple CPU cores seemingly to run in parallel but in fact each CPU core interacts with one GPU immediately one after another. For more details about this advanced computational technique and the R package, see our OSF site (https://osf.io/p4pdh). For details regarding the experimental design of the data set, see Holmes et al., (2016).

Results

We use DIC to assess whether the PLBA model *1f* accounts for the data better than the LBA model. Note that it is important in this comparison that the same PDA methods (i.e., 1,048,576 model simulations with kernel bandwidth 0.01 s) be used for both models because, as shown earlier, changes in method can change the absolute DIC. We chose 2^{20} model simulations, because the results in the PLBA simulation studies suggest that it is possible to identify some post-switch parameters with this setting and it is relatively more efficiently than using 2^{27} model

simulations. We calculated DIC separately for each participant and summed over them to assess the overall fits.

Table 1 shows that although most participants show smaller DIC in the PLBA fits than in the LBA fits (20/31), only two participants reach difference larger than ten. Seven participants show no difference and four had a larger DIC in the PLBA than in the LBA fits. When aggregated across the participants, the total DIC for the LBA and PLBA *1f* models are, respectively, 31,571 and 31,530 with a difference of 41, favoring the PLBA model. The hierarchical model fits reported by Holmes et al. (2016) found a DIC for the LBA and the PLBA models of 1,007 versus 926, a larger advantage of 81 favoring the PLBA *1f*.

[Insert Table 1 Here]

Figure 4 and Figure 5 show the quality of fit of the RT distributions for the pre-switch and post-switch correct choices separately for each participant. These figures were produced by firstly constructing the data histograms. We then randomly sampled 100 sets of PLBA parameters without replacement from the posterior distribution to simulate 100 posterior predictive data sets. Each set is represented by one gray line in the figures, resulting in gray ribbons. In general, the PLBA *1f* model fit the pre-switch correct responses closely, whereas when a correct response is defined by matching the stimulus direction after switch, the model fit to the data fit is less closely.

[Insert Figure 4 and Figure 5 Here]

Best Practice of the PDA Method

We conducted three LBA simulation studies to examine the effectiveness and accuracy of the pPDA in recovering data generating parameters (i.e., parameter-recovery studies, see Heathcote, Brown, & Wagenmakers, 2015) and provided guideline of applying PDA. To

foreshadow, these studies suggest, firstly, that one must take noise introduced by PDA into account in making inferences based on PDA model fits. For example, model selection and parameter uncertainty calculated from PDA fits quantitatively differ from those calculated using analytic likelihood functions. Secondly, whenever possible, one should conduct model simulations at the scale of millions to construct simulated PDFs. Thirdly, in the case of fitting RT data, one should use a kernel bandwidth of approximately 0.01 seconds, as it provides a good balance between estimation bias and variance. Lastly, in order to avoid profound decrements in MCMC efficiency, one must modify the standard practice of reusing likelihoods calculated on earlier iterations.

The studies examined several questions, regarding how to best use PDA in Bayesian computation. We then summarized the results as PDA guidelines. All simulation studies used truncated normal distributions (Robert, 1995) as priors for the upper bound of start point *A*, the distance above that to the threshold, *B*, the mean "correct" drift rate, μ_{v_c} (i.e., the rate for the accumulator that matches the stimulus), and the mean error drift rate, μ_{v_e} (i.e., the rate for the mismatching accumulator). The priors for *B, A*, μ_{v_c} and μ_{v_e} were truncated at the zero. The prior distribution for the non-decision time is uniform, with a lower bound of either 0.1 or 0.01 seconds and an upper bound of 1 second.

Study I

The benefit of scaling up model simulations towards aysmptotic level has been stated theoretically (Parzen, 1962; van Zandt, 2000). Holmes (2015) investigated the influence of the numbers of model simulations in practice in three cases, 5,000, 10,000 and 40,000, for one hundred independent fits. Figure 5(a) in Holmes (2015) also demonstrated a case where the simulated PDF becomes very close to real PDF suggesting that one million model simulations

comes close to asymptotic accuracy. Our first study, taking advantage of the parallel GPU computation implemented in pPDA, further investigates this case. We ran one hundred independent fits and systematically compared the result using one million model simulations with the results with lesser numbers of model simulations.

Method

This study set μ_{v_c} and μ_{v_e} to 1.0 and 0.25 and used the same A, b and t_0 as the replication study. This set of parameters results in data with around 63% accuracy and generates slower errors than the parameters in the replication study, where the accuracy is about 70%. We conducted three parameter recovery studies, each with a different number of model simulations. The first used 8,192, less than Turner and Sederberg (2014), so we expected worse parameter recovery. The second and third used 16,384 and 1,048,576. We expected the third study would show asymptotically unbiased and consistent estimations, as is the case for density estimation (Parzen, 1962; Van Zandt, 2000). We also conducted a control study, using the analytic likelihood. This study also used the analytic likelihood function to fit data with 500 trials per condition.

Results

Figure 6 shows a clear influence of the numbers of model simulations on parameter estimation, highlighting three findings. Firstly, as in the replication study, PDA introduces additional estimation error, which is markedly greater for 8,192 and 16,384 (red & blue lines) than for 1,048,576. Second, an increase in the simulation numbers decreases the approximation noise. Third, when the number of model simulations is 1,048,576, PDA posterior distributions are almost the same as analytic posterior distributions in the parameters, μ_{v_c} and μ_{v_e}, and very close in the parameters, B, A, and t_0. Figure 7 shows the PDA method has almost identical fits to

response proportions and RT quantiles as the analytic likelihood when the simulations are above one million. The DIC values are 1820, 1772, and 1621 for the three PDA estimates with increasing model simulations and 1556 for the analytic likelihood estimate. Hence, even with over one million model simulations the inflation in DIC is sufficient to mandate against comparison with DIC based on analytic likelihoods, as a DIC difference of 10 or greater is considered large.

[Insert Figure 6 & Figure 7 Here]

Study II

As noted in van Zandt (2000), the number of model simulations exerts its influence through the kernel function, which is modulated by the kernel bandwidth. This study investigates the role of the kernel bandwidth in association with the number of model simulations. There are a variety of methods that have been proposed to find an optimal bandwidth in KDE (Chiu, 1991; Goldenshluger & Lepski, 2011; Silverman, 1986). Silverman's plug-in method is perhaps one of the most widely used and intuitive methods, but it may not be always optimal. We investigate the optimal bandwidths for RT data for relatively simple decisions, which spanning the range from 0.1 second to at most a couple of seconds. The following two equations for the bias and variance of KDE reproduced from Holmes (2015; see also Silverman, 1986, p38-39) provide a general guidance for the selection of testing bandwidths.

$$Bias\left(\hat{f}(y)\right) \approx \frac{h^2}{2} f''(y) M_2(K) \qquad (8)$$

$$Var(\hat{f}(y)) \approx \frac{1}{N_s h} f(y) ||K||_2 \qquad (9)$$

Here h and N_s denote the KDE bandwidth and the number of model simulations, respectively. $M_2(K)$ denotes the second moment of the kernel function, K, and $||K||_2$ denotes its Euclidean

distance (i.e., L^2 norm), y is the empirical data to be estimated, so $\hat{f}(y)$ is the estimated PDF and $f''(y)$ is the second derivative of the likelihood function.

The aim for an optimal bandwidth is to jointly minimize bias and variance of density estimates. Select a small bandwidth reduces bias, but as shown in the variance equation, a decrease in the bandwidth causes an increase in the variance. Again, we resolve this dilemma by using many model simulations. Our general approach is to choose the bandwidth such that bias is minimal (usually in the range of one to tens of milliseconds), and subsequently chose the number of simulations large enough to reduce the variance sufficiently. The latter condition is greatly facilitated by use of the GPU since it makes possible the use of large N_S.

Method

We examine the bandwidth selection problem, testing 13 different potential bandwidths for choice RT data (on the seconds scale): 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9 across three different numbers of model simulations, 5,000, 10,000, and 1,048,576. The first two cases were conducted using only CPU, because when the numbers of model simulation are less than 2^{15} (= 32,768), CPUs tends to outperform GPUs. Each of the 39 combinations were conducted independently 100 times each. We expected that the influence of bandwidth on variance should become negligible when the number of model simulations is very large and that the influence should become apparent when the number of model simulation is small. This study used pPDA to fit data sets with 500 trials per condition.

The reason we tested absolute bandwidths, instead of using, for example, Silverman's (1986) method is that such automatic methods do not fare well with simulated RT data due to occasional very large values. The number of slow simulated RTs is relatively small when the number of model simulations is below a few dozen thousands. When we raise the number of

model simulations to more than one million, the number of slow RTs increases. This characteristic renders Silverman's method less ideal, because it then often suggests a very large bandwidth which is clearly sub-optimal. Although in Silverman's method one can choose adjusted inter-quantile ranges, instead of standard deviation, as its bandwidth suggestion, this method does not mitigate the problem because when the number of model simulation is very large, the slow RTs become very slow. Instead, given we know the typical scale of our data from previous literature, we selected a wide range possible bandwidth that might yield optimal PDA performance in fitting RT data.

Results

Figure 8 presents the distribution over 100 replications of the root mean squared estimation errors (RMSEs) averaged over parameters and Figure 9 shows the width of 95% credible intervals for each parameter averaged over. When bandwidth is greater than 0.1 seconds, biases (Figure 8) and variances (Figure 9) increase strongly. This is true across all three simulations sizes, although using more than one million model simulations mitigates the problem. For 5,000 and 10,000 model simulations, the biases increase gradually when bandwidth is less than 0.01 seconds, but decrease slightly in the case of over one million model simulations. The variances of *A, B,* and t_0 are at a minimum for 0.1 seconds for 5,000 and 10,000 model simulations whereas the minimum for μ_{v_c} and μ_{v_e} is at 0.01 seconds. For over one million model simulations variance is fairly constant for 0.1 seconds or less, except for μ_{v_c} and μ_{v_e} where it is higher for 0.1 seconds and low and constant for smaller bandwidths.

[Insert Figure 8 & Figure 9 Here]

In summary, this study suggests that a bandwidth between 0.01 to 0.1 seconds provides the best compromise for minimizing bias and variance, with larger numbers of model simulations reducing the effect of this choice and pushing the best value a little lower.

Study III

The recalculation method introduced by Holmes (2015) is a new specific adaption for applying PDA in Bayesian MCMC using a Metropolis-Hastings sampler. This method, although imposing an additional computational burden by requiring the likelihood of accepted samples to be sometimes recalculated, prevents the problem of chain stagnation. It remains an unexplored question regarding how different recalculation intervals affect the performance of the Bayesian MCMC. This study examines this question. We recalculated the likelihoods of accepted samples every 2, 4, 8, 16, 32, 64, 128, and 256 consecutive iterations.

Method

We generated 10,000 responses from the LBA model, using the same parameters as in the replication study. We only used the crossover operator (Ter Braak, 2006) in this study as it has been mathematically proven (Hu & Tsui, 2010) to be able to reach target distributions and draws MCMC samples from there, whereas similar works have yet to be done for other more recent genetic, such as migration, operators, although they do have great utility during the burn-in phase in finding target distributions. We used PDA with 1,048,576 model simulations and a bandwidth of 0.01 seconds.

Results

We deployed thirty chains, discarded the first 512 samples and kept every second sample for the next 512 iterations so that the way chains move would be evident. Results are plotted in Figure 10 in terms of sampled posterior log-likelihood values for each chain. Figure 10 presents

only three recalculation intervals, 4, 16 and 64, with a longer x-axis to show increasing chain stagnation (i.e., periods of constant likelihood) as recalculation became more infrequent. An initial finding suggests chain stagnation is a problem of efficiency, rather than a problem that prevents chains from converging.

[Insert Figure 10 & Figure 11 Here]

We then ran further iterations, adjusted thinning intervals, and stopped the model fits until the chains become stationary, well-mixed, and have at least 512 "effective" samples (i.e., adjusted for the effects of autocorrelation). Figure 11 shows there is an increasing bias towards higher and less variable likelihoods as the recalculation intervals increase, at least for intervals greater than 4. Although this has very little effect in terms of parameter bias it has a marked effect of reducing the estimates of uncertainty provided by credible intervals, which would cause spurious overconfidence in the precision of the estimates. It also causes DIC to decrease, with the higher average indicating a spuriously better fit and the reduced variance a spuriously decease in model complexity. For intervals of 2 and 4 results appeared to be reasonably stable.

Discussion

Taken together, the simulation studies provide guidelines for applying PDA when fitting RT data. First, the PDA will produce estimation noise due to its approximate nature. Second, the approximation noise can be gradually resolved by conducting more model simulations. This improvement, however, is not homogeneous across model parameters. Comparing to using the analytic likelihood, PDA with over one million model simulations gives similarly good estimations for the drift rates, but results in some differences in the other parameters. Third, an optimal bandwidth in PDA for fitting the RT data is around 0.01 seconds. Fourth, it best to recalculate as frequently as possible given sufficient computational resources, and at least every

4^{th} iteration, as suggested by Holmes (2015). Finally, DIC and credible intervals cannot be compared between analytic and PDA results, even those based on one million or more simulations, and between PDA results with different numbers of simulations or with different recalculation intervals.

Performance Profiling

We conducted three performance profiling, comparing the pPDA with the conventional CPU-based PDA. First, we measured the time it takes to fit the LBA model to simulated data. pPDA addresses two computational bottlenecks: (i) drawing many model simulations and (ii) synthesizing these simulation samples into a likelihood (Holmes, 2015, p.22). The first performance profiling should reveal the difference with regards to the two improvements. Second performance profiling compared the influences of the numbers of model simulations on the computation time of pPDA with that of CPU-based PDA. In the second performance profiling, we simulated 10,000 trials, based on the PLBA model with the same parameters as in the third PLBA simulation study (see Appendix), and calculated the 10,000 PLBA probability densities. We tested eight different numbers of model simulations, 2^{14}, 2^{15}, 2^{16}, 2^{17}, 2^{18}, 2^{19}, 2^{20} and 2^{21}. Each case was done independently for 100 times. The third performance profiling, similar with the second one, measured the time for calculating 10,000 PLBA probability densities, but with only pPDA method and 16 different numbers of model simulations, 2^{14}, 2^{15}, 2^{16}, 2^{17}, 2^{18}, 2^{19}, 2^{20}, 2^{21}, 2^{22}, 2^{23}, 2^{24}, 2^{25}, 2^{26}, 2^{27}, 2^{28}, and 2^{29}. The aim of the third profiling was to determine to what extent allocating many GPU memories becomes a computational bottleneck.

To make a fair comparison, the regular PDA was done by re-coding the MATLAB PDA from Holmes (2015) in C++ and including it our R package, *ggdmc*. The identical method of

software packaging allows us to compare the two methods in similar computational environments, so the performance difference should mostly be attributable to GPU computation.

Method

In the first comparison, we timed pPDA in two computational environments. The first was a desktop computer, equipped with an Intel® Core ™ i7-5930K six-core CPU, which runs at a 3.50 GHz clock rate with a capacity for computing 12 processes in parallel. This desktop computer was equipped with a Nvidia Tesla K20 GPU and a GeForce 980 GPU. Note that because we code pPDA in CUDA C, it works only with Nvidia GPUs. Second, we timed CPU-based PDA on four identical virtual machines. Each machine was configurated with an Intel 12-core CPU, running at a 2.6 GHz clock rate and a Nvidia Tesla K80 GPU. We conducted 200 Bayesian fits on the empirical data (Holmes et al, 2016), using a 5-parameter LBA model (start-point variability, a decision threshold, a non-decision time, and correct and error mean drift rates). The first 100 fits used pPDA and the others used CPU-based PDA. Note the strength of GPU computing is its ability to conduct massively parallel computations. At a moderate number of parallel computations, GPU computing usually does not outperform CPU parallel computing because the computation speed for each GPU core is slower than that of CPU core. Hence, both methods draw 1,048,576 model simulations to synthesize PDFs. All model fits ran 512 iterations using a mixture of the *crossover* (Ter Braak, 2006) and the *migration* operators (Turner et al., 2013) and then ran another 1024 iterations, using only the *crossover* operator. Both methods recalculated likelihood every four iterations.

In the second and third comparisons, we timed the performance on one of the virtual machines, using eight different model simulations. The 10,000 simulated trials were generated

from the PLBA model, $A = 0.75, B = 0.25, \mu_{v_1} = 2.5, \mu_{v_2} = 1.5, \mu_{w_1} = 1.2, \mu_{w_2} = 2.4, rD = 0.1,$ and $t_0 = 0.2$. This was done separately for GPU and CPU.

[Insert Table 3 & Figure 12 Here]

Results

As shown in Table 3, on average, pPDA finishes one Bayesian model fit in 43 hours, which was almost 3 times faster than the CPU-based PDA [120 hours, $t(99) = 40.34$, $p < .0001$]. Table 3 **Error! Reference source not found.**shows the fitting times of CPU-based fits (SD = 19.25 hours) are also more variable than that of pPDA (SD = 0.34 hours). The maximum time for CPU-based PDA to finish a fit was 143 hours, but sometimes it finished in only 84 hours, whereas all the GPU computations were around 42 or 43 hours. This is foreseen because the GPU uses a block of 32 threads (the default thread size in *ppda*) to conduct model simulations in parallel, but the CPU (one core) PDA drew each model simulation sequentially. The former drew model simulations in a relatively homogeneous computational setting (e.g., CPU, GPU temperatures, and available RAM etc.), but the latter drew each model simulation in a slightly different setting. This suggests that with respect to the computation time, pPDA is also more predictable.

Figure 12 shows the times for calculating PLBA probability densities. There are four key findings here. Firstly, the computation time of CPUs increase linearly (on a $\log_{10}$ scale) with the number of model simulations. Second, the time it takes for CPUs to calculate 10,000 PLBA densities using 16,384 model simulations (median = 40 ms) is similar with that of GPUs, using 262,144 model simulations (median = 39 ms), speedup by a factor of 16. Third, GPUs scale better than CPUs as the numbers of model simulations increase. It takes more than 4 seconds (median = 4220 ms) for a CPU to calculate 10,000 PLBA densities with 2,097,152 model

simulations, but just 113 ms for a GPU to do so, a factor of 37. Lastly, when the number of model simulations is up to 2^{23} (over eight million), the burden of handling large GPU memory spaces gradually manifest. When the numbers of model simulation are more than 2^{25}, the computation time of GPUs become also linear (on a $\log_{10}$ scale) with the number of model simulations.

General Discussion

Parallel PDA makes approximate Bayesian computation practical in fitting cognitive models where likelihood functions are often intractable. PDA has its roots in KDE (Parzen, 1962; Silverman, 1986; Van Zandt, 2000), a method estimating probability densities from Monte Carlo simulated data. By harnessing heterogeneous GPU/CUDA and CPU/ C++ programming model, pPDA overcomes the main obstacle to this approach, the intense computations required to obtain sufficiently large simulations for each of the many iterations required by Markov Chain Monte Carlo methods.

The goal of this paper has been to provide a set of practical tools and guidelines to conduct Bayesian computation on intractable evidence accumulation models efficiently. We used CUDA and Armadillo C++ to implement pPDA in an R package, *ppda*. Although CUDA limits this approach to Nvidia hardware, it allows the user an easy path to harness massively parallel GPU computation via the accessible R language. The challenge others have encountered (e.g., Verdonck, Meers, & Tuerlinckx, 2015) is to transfer synthesized data occupying a huge amount of GPU memory back to CPU side for handling. Instead of choosing the strategy of fine-tuning CUDA stream scheduling, we opt for a smarter strategy, conducting parallel reduction, which is a parallel algorithm, not merely a CUDA programming technique. This strategy makes use of different GPU memory in a different context, helping us enhance efficiency greatly, as

documented in Harris (2007). We use parallel reduction to extract key statistics from the synthesized data inside GPU memory, and transfer only these key statistics to CPU memory, rather than transferring all synthesized data out. Therefore, profiling memory usage and its impact on performance is simply not applicable to our case, because our pPDA transfers very tiny amounts of memory in and out of GPU. This paper thus removes one of the many of the impediments associated with modeling.

We conducted a series of simulation studies fitting the LBA model (Brown & Heathcote, 2008), which has a tractable likelihood against which we can benchmark pPDA performance. The results suggest one should use GPUs to synthesize the simulated histogram with as many, such as over one million, model simulations as possible, set a bandwidth, at least smaller than 0.01 second, and enforce a modification of standard Metropolis methods suggested by Holmes (2015) recalculating the likelihood of previously accepted samples. We then went on fitting the PLBA model to empirical data (Holmes et al., 2016). This example, together with recent PDA applications on other complex models (Holmes & Trueblood, 2018; Miletić, Turner, Forstmann, & van Maanen, 2017; Trueblood et al., 2018), demonstrate one can apply pPDA on fitting intractable cognitive models. In the following, we discuss the issues regarding problems for Markov Chain Monte Carlo methods caused by PDA, the limitations of our approach, and its future development.

Sampling Problems Caused by PDA

One problem in applying PDA in Bayesian computation is *likelihood inflation* (Holmes, 2015). This problem causes spuriously large likelihoods due to noise in PDA estimates. Likelihood inflation results in chain stagnation, as other plausible proposals are rejected in favor of the spuriously likely samples, and so the chain remains unchanging. Fortunately, it can be

resolved by recalculating the likelihoods of accepted samples but will double computational cost if done on every occasion. We investigated mitigation of this extra cost by performing recalculation less often. We found that long recalculation intervals, although slowing down chain mixing, do not prevent Markov chains from reaching convergence. However, although saving computation, such intervals spuriously reduce variability, and so can produce an overlay optimistic picture of the level of certainty in estimates.

Limitation and Future Development

PDA, although applying to a wide range of cognitive models, does not solve all modeling obstacles, and introduces some pitfalls, with computation time and approximation noise being the most prominent. The purpose of this paper is to both make users aware of such pitfalls and to make available a highly efficient parallel implementation that provides methods to address them.

Because the developmental landscape of GPU and CUDA libraries is changing rapidly, we cannot make clear predictions regarding the influences of future GPUs and CUDA libraries on *ppda*. Here, we provide recommendations based only on the four types of GPU we have tested: Tesla K80, Tesla K20, GeForce GTX 980, and GeForce GT 720M. The former two GPUs are designed for servers, the third is for desktop computers, and the last is for notebook computers. All return correct results, although each has a different computational speed. In general, the more expensive a GPU, the faster its CUDA cores calculate. This roughly matches the versioning system, *Compute Capability* (Nvidia, 2018, p.15). For example, we found Tesla K80 with a 3.7 *Compute Capability* calculates faster than Tesla K20, which has a 3.5 *Compute Capability*. However, there are other factors to consider. First is the size of on-board memory. One Tesla K80 card ships with two GPUs, each of them equipping with 12 GB memory. In contrast, Tesla K20 and GeForce GTX 980 come with one GPU and less than 5 GB memory.

This directly affects how many model simulations can be accommodated. pPDA on Tesla K80 allows up to almost one billion model simulations, but this is not possible on the other GPUs without dividing one job among multiple GPUs. Secondly, the maximum number of parallel threads in a computing block also affects speed. The earlier GeForce GT720M GPU allows only 512 maximum parallel threads in a block. We set a default launching block size at 32 parallel threads to accommodate these GPUs. Recent GPUs allow 1024 maximum threads, so earlier GPUs, although returning correct results, will require more time to conduct PDA. In more recent GPUs one might further reduce computation times by setting a larger block with for example 1024 parallel threads. This can be easily done by setting, *nthread = 1024,* in *ppda*'s function calls. However, we have not yet thoroughly tested the influence of difference block sizes on computation times, and CUDA programming involves many other intricacies related to the design of Nvidia GPU and to parallel programming methods, such as *wrap divergence* (Cheng, Grossman, & McKercher, 2014, p. 82). In summary, a good and recent GPU, such as GeForce GTX 980, for a desktop PC might be a wise choice for someone who would like to balance financial cost and fast computation with our package.

Conclusion

pPDA, which is built on the foundation of previous KDE and PDA developments (Parzen, 1962; Van Zandt, 2000; Turner & Sederberg, 2014; Holmes, 2015), equips researchers with the unprecedented ability to conduct many model simulations efficiently. Its open source implementation, *ppda*, allows the user to add a new model by writing a CUDA kernel function for model simulations and linking it to the PDA routines. Hence, *ppda* enables researchers to explore other process models with only small investment (e.g., a PC equipped with a fast multicore CPU and Nvidia GPUs). pPDA's development adheres to the strict R package

standard, making *ppda* and its source codes accessible. In addition, we provide guidelines regarding how to apply PDA in Bayesian computation, to make *ppda* a computational tool, allowing future researchers to explore a wide range of questions in cognitive process models.

Reference

Ball, K., & Sekuler, R. (1982). A specific and enduring improvement in visual motion discrimination. *Science*, 218(4573), 697-698.

Beaumont, M. A. (2010). Approximate Bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution, and Systematics*, 41(1), 379-406.

Brooks, S., Gelman, A., Jones, G., & Meng, X. L. (2011). *Handbook of Markov Chain Monte Carlo*. CRC press.

Brown, S. D., & Heathcote, A. (2008). The simplest complete model of choice response time: linear ballistic accumulation. *Cognitive Psychology*, 57(3), 153-178.

Cheng, J., Grossman, M., & McKercher, T. (2014). *Professional CUDA C programming*. Indianapolis, Indiana: John Wiley & Sons.

Chiu, S.-T. (1991). Bandwidth selection for kernel density estimation. *The Annals of Statistics*, 19(4), 1883-1905. Retrieved from http://www.jstor.org/stable/2241909

Cisek, P., Puskas, G. A., El-Murr, S. (2009). Decisions in changing conditions: the urgency-gating model. *Journal of Neuroscience*, 29(37), 11560-11571

Dawson, M. R. (1988). Fitting the ex-Gaussian equation to reaction time distributions. *Behavior Research Methods, Instruments, & Computers*, 20(1), 54-57.

Gelman, A. (2014). *Bayesian Data Analysis*. Boca Raton: CRC Press.

Dutilh, G., Annis, J., Brown, S. D., Cassey, P., Evans, N. J., Grasman, R. P. P. P., Hawkins, G. E., Heathcote, A., Holmes, W. R., Krypotos, A.-M., Kupitz, C. N., Leite, F. P., Lerche, V., Lin, Y.-S., Logan, G., Palmeri, T. J., Starns, J. J., Trueblood, J. S., van Maanen, L., van Ravenzwaaij, D., Vandekerckhove, J., Visser, I., Voss, A., White, C. N., Wiecki, T. V., Rieskamp, J., & Donkin, C. (2018). The quality of response time data inference: A

blinded, collaborative assessment of the validity of cognitive models. *Psychonomic Bulletin & Review*, 1-19. doi: 10.3758/s13423-017-1417-2

Goldenshluger, A., & Lepski, O. (2011). Bandwidth selection in kernel density estimation: Oracle inequalities and adaptive minimax optimality. *The Annals of Statistics*, 39(3), 1608-1632. doi:10.1214.11-AOS883

Gureckis, T. M., & Love, B. C. (2009). Learning in noise: Dynamic decision-making in a variable environment. *Journal of Mathematical Psychology*, 53(3), 180-193.

Harris, M. (2007). Optimizing parallel reduction in CUDA. Retrieved from http://docs.nvidia.com/cuda/samples/6_Advanced/reduction/doc/reduction.pdf

Heathcote, A. (2004). Fitting Wald and ex-Wald distributions to response time data: An example using functions for the S-PLUS package. *Behavior Research Methods, Instruments, & Computers*, 36(4), 678-694.

Heathcote, A., Lin, Y.-S., Reynolds, A., Strickland, L., Gretton, M., & Matzke, D. (2018). Dynamic Models of Choice. *Behavior Research Methods*. submitted for publication.

Hoffman, M. D., & Gelman, A. (2014). The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1), 1593-1623.

Holmes, W. R. (2015). A practical guide to the probability density approximation PDA with improved implementation and error characterization. *Journal of Mathematical Psychology*, 68-69, 13-24.

Holmes, W. R., & Trueblood J. S. (2018). Bayesian analysis of the piecewise diffusion decision model. *Behavior Research Methods*, 50(2), 730-743.

Holmes, W. R., Trueblood, J. S., & Heathcote, A. (2016). A new framework for modeling decisions about changing information: The Piecewise Linear Ballistic Accumulator model. *Cognitive Psychology*, 85, 1-29.

Hu, B., & Tsui, K.-W. (2005). Distributed Evolutionary Monte Carlo with Applications to Bayesian Analysis. Madison: Department of Statistics, University of Wisconsin - Madison.

Hu, B., & Tsui, K.-W. (2010). Distributed Evolutionary Monte Carlo for Bayesian Computing. *Computional Statistics and Data Analysis*, 54(3), 688-697. doi:10.1016/j.csda.2008.10.025

Luce, R. D (1986). *Response times*. New York: Oxford University Press.

Matzke, D., & Wagenmakers, E.-J. (2009). Psychological interpretation of the ex-Gaussian and shifted Wald parameters: A diffusion model analysis. *Psychonomic Bulletin & Review*, 16(5), 798-817.

McClelland, J. L. (1979). On the time relations of mental processes: An examination of systems of processes in cascade. *Psychological Review*, 86(4), 287-330.

Miletić, Turner, Forstmann, & van Maanen, (2017). Parameter recovery for the Leaky Competing Accumulator model. *Journal of Mathematical Psychology*, 76, 25-50.

Neal, R. M. (1994). An improved acceptance procedure for the hybrid Monte Carlo algorithm. *Journal of Computational Physics*, *111*(1), 194-203.

Nvidia. (2018). CUDA C programming guide PG-02829-001_v9.1 | March 2018. Retrieved 20 Apr 2018 from https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

Parzen, E. (1962). On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3), 1065-1076.

Palestro, J. J., Sederberg, P. B., Osth, A. F., van Zandt, T., & Turner, B. M. (2018). *Likelihood-Free Methods for Cognitive Science*. Cham: Springer International Publishing

R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, 85(2), 59-108.

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Computation*, 20(4), 873-922.

Robert, C. P. (1995). Simulation of truncated normal variables. *Statistics and Computing*, 121-125. doi:10.1007/BF00143942

Roberts, G. O., & Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16(4), 351-367.

Sanderson, C., & Curtin, R. (2016). Armadillo: a template-based C++ library for linear algebra. *The Journal of Open Source Software*, 1(2).

Schwarz, W. (2002). On the convolution of inverse Gaussian and exponential random variables. *Communications in Statistics Theory and Methods*, 31(12), 2113-2121.

Silverman, B. W. (1982). Algorithm AS 176: Kernel density estimation using the fast Fourier transform. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(1), 93-99. doi:10.2307/2347084

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall.

Sisson, S. A., & Fan, Y. (2010, Jan). Likelihood-free Markov chain Monte Carlo. arXiv:1001.2058 [stat]. Retrieved from http://arxiv.org/abs/1001.2058

Smith, P. L. (2016). Diffusion theory of decision making in continuous report. *Psychological Review*, 123(4), 425.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & Van Der Linde, A. (2002, Oct). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B Statistical Methodology*, 64(4), 583-639.

Ter Braak, C. J. (2006). A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: easy Bayesian computing for real parameter spaces. *Statistics and Computing*, 16(3), 239-249.

Tsetsos, K., Usher, M., & McClelland, J. L. (2011). Testing multi-alternative decision models with non-stationary evidence. *Frontiers in Neuroscience*, 5.

Thura, D., Beauregard-Racine, J., Fradet, C.-W., & Cisek, P. (2012). Decision-making by urgency gating: Theory and experimental support. *Journal of Neurophysiology*, 108(11), 2912-2930.

Trueblood, J. S., Holmes, W. R., Seegmiller, A. C., Douds, J., Compton, M., Szentirmai, E., ... & Eichbaum, Q. (2018). The impact of speed and bias on the cognitive processes of experts and novices in medical image decision-making. *Cognitive Research: Principles and Implications*, *3*(1), 28.

Turner, B. M., & Sederberg, P. B. (2012). Approximate Bayesian computation with differential evolution . *Journal of Mathematical Psychology* , 56(5), 375-385.

Turner, B. M., & Sederberg, P. B. (2014). A generalized, likelihood-free method for posterior estimation. *Psychonomic Bulletin & Review*, 21(2), 227-250.

Turner, B. M., Sederberg, P. B., Brown, S. D., & Steyvers, M. (2013). A method for efficiently sampling from distributions with correlated dimensions. *Psychological methods*, 18(3), 368.

Usher, M., & McClelland, J. L. (2001). The time course of perceptual choice: the leaky, competing accumulator model. *Psychological review*, 108(3), 550.

Van Zandt, T. (2000). How to fit a response time distribution. *Psychonomic Bulletin & Review*, 7(3), 424-465.

Verdonck, S., Meers, K., & Tuerlinckx, F. (2016). Efficient simulation of diffusion-based choice RT models on CPU and GPU. *Behavior Research Methods*, 48(1), 13-27.

Tables

Table 1. DIC values of the PLBA model 1f and the LBA model.

	S125	S127	S128	S131	S132	S135	S136	S137	S141	S142
LBA	861	325	1096	1286	664	947	-323	987	224	923
PLBA	860	326	1095	1288	663	947	-324	991	222	921
	S144	S145	S146	S149	S151	S152	S153	S154	S157	S158
LBA	821	749	1512	504	864	1766	1829	1335	1279	1790
PLBA	812	747	1511	500	861	1766	1815	1333	1278	1801
	S159	S160	S163	S164	S165	S166	S167	S168	S169	S170
LBA	1752	1253	1670	2358	1358	575	166	1628	566	-422
PLBA	1751	1254	1665	2351	1357	573	166	1628	567	-421
	S171									
LBA	1229									
PLBA	1227									

Note: S stands for subject/participant.

Table 2. Point estimates and Bayesian credible intervals.

	A	B	μ_{v_1}	μ_{v_2}	μ_{w_1}	μ_{w_2}	rD	t_0
True values	0.750	0.250	2.500	1.500	1.200	2.400	0.100	0.200
2.5% Estimate	0.722	0.167	2.363	1.315	1.026	1.870	0.107	0.206
50% Estimate	0.774	0.199	2.494	1.449	1.593	2.270	0.128	0.216
97.5% Estimate	0.831	0.237	2.627	1.588	2.117	2.687	0.156	0.225

Table 3. Performance profiling (in hours).

	Min.	*1st Qu.*	*Median*	Mean	3rd Qu.	Max
Intel CPU	83.61	114.63	130.68	120.17	135.56	143.31
Nvidia K20	41.34	42.43	42.49	42.50	42.61	43.85

Figures

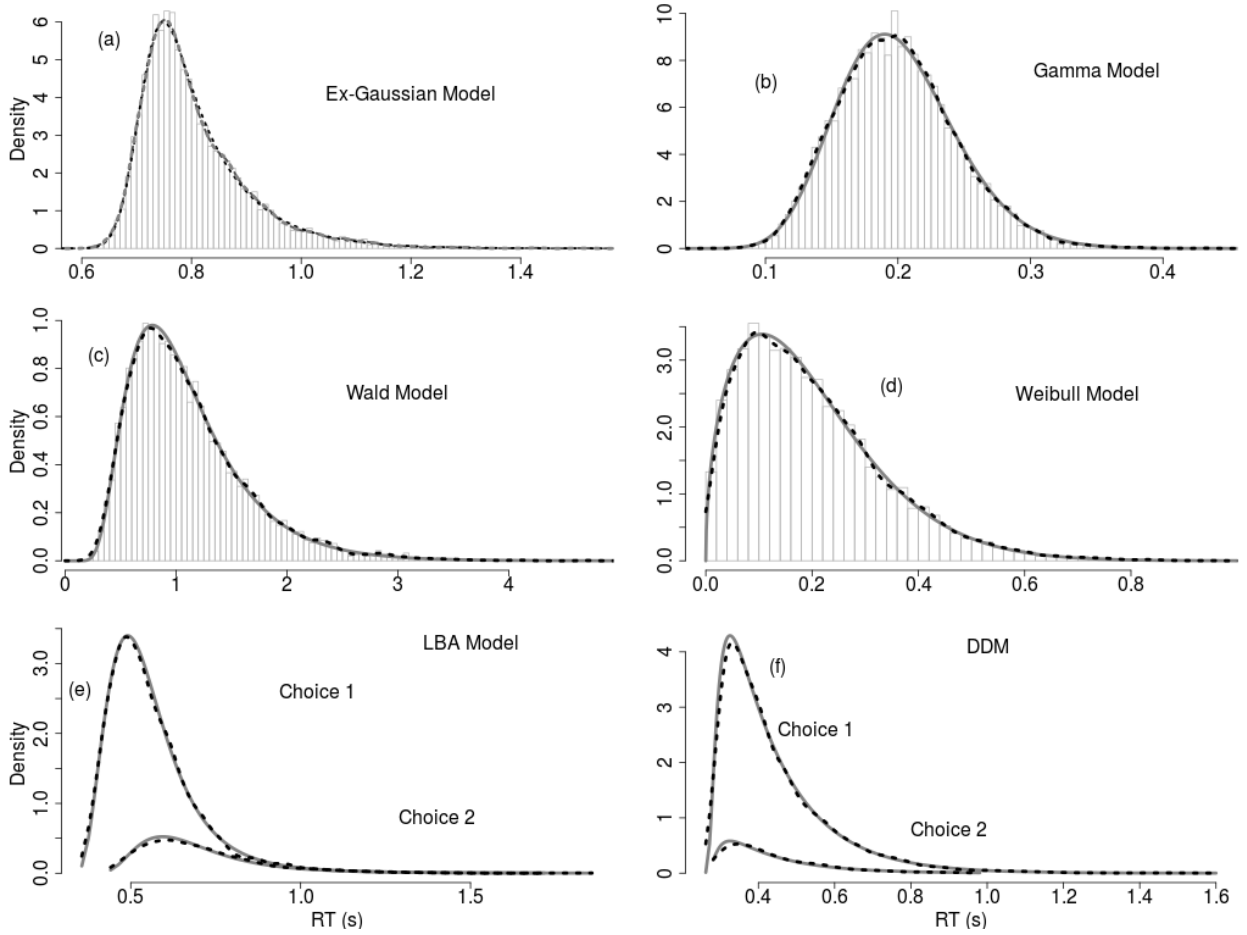

Figure 1. PDA fits to six cognitive models: ex-Gaussian, gamma, Wald, Weibull and linear ballistic accumulation (LBA) models and diffusion-decision model (DDM). The gamma, Wald and Weibull distributions have subtracted out non-decision times. The histograms show the synthesized probability density functions. In the panel (a), three methods, analytic solution, KDE, KDE-FFT, are represented respectively by a dark dotted, a gray dotted line and a dashed gray line. The three lines are almost completely overlapping, so it is hard to see individual line. All the other panels used solid gray and dotted lines to represent the fits of analytic solution and PDA. For the gamma model, we presume an accumulator takes twenty steps (shape = 20) to reach a decision with 0.01 second as the step size. The Weibull model used a shape parameter of 1.5 and a scale parameter of 0.22. The DDM used a drift rate of 2, a boundary separation of 1 and a t_0 of 0.25 seconds. The LBA model data were generated with the following parameters: upper bound for the starting evidence, $A = 0.5$, response threshold, $b = 1$, non-decision time, $t_0 = 0.25$, mean drift rate for correct response, $\mu_{v_c} = 2.4$, (i.e., Choice 1) and error response $\mu_{v_e} = 1.2$ (i.e., Choice 2), and their standard deviations, $\sigma_{v_c} = 1.0$ and $\sigma_{v_c} = 0.6$. We generated 1,000 choice RT data from the LBA and diffusion models. All six cases used 10,000 model simulations to synthesize PDFs.

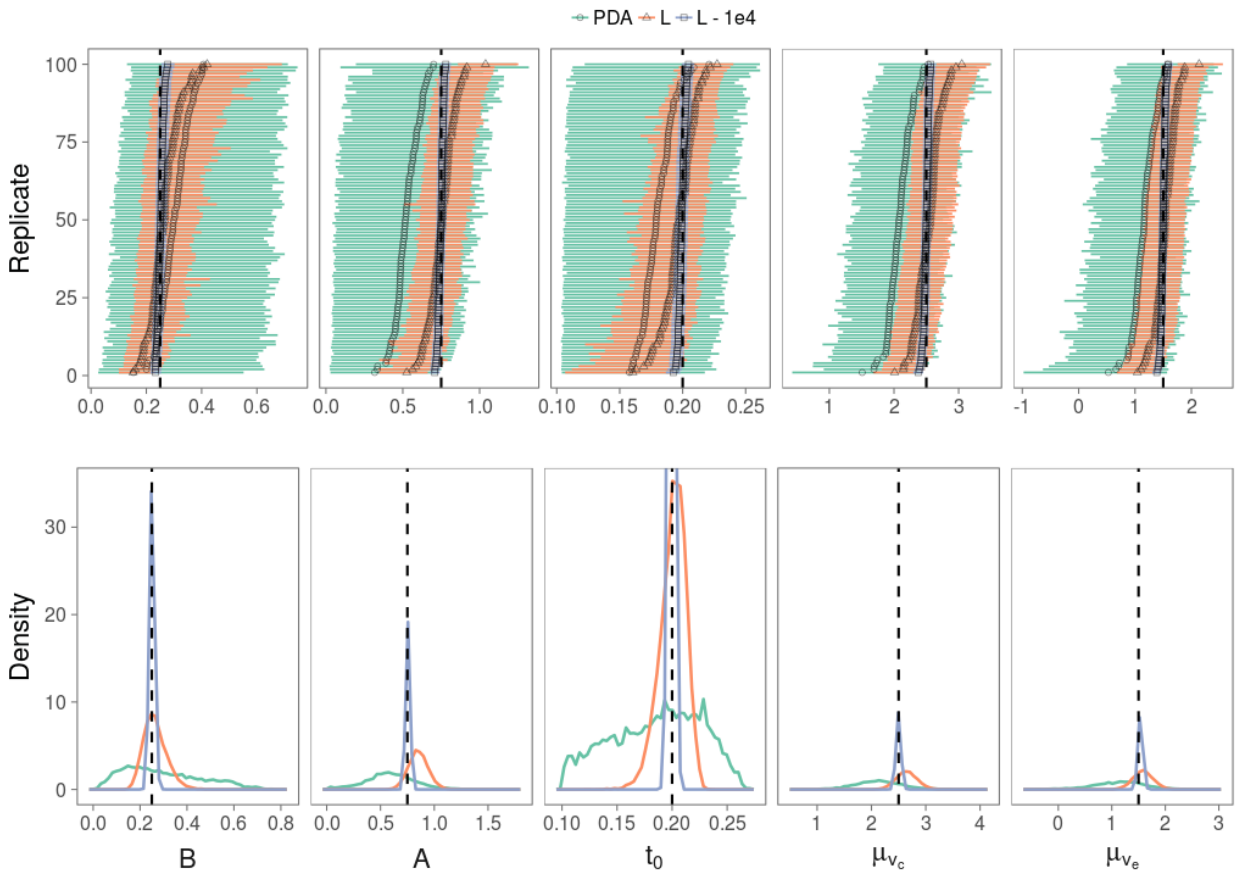

Figure 2. Marginal posterior distributions and 95% Bayesian credible intervals. The dashed lines show true parameter values: $b = 1.0, A = 0.75, (B = 0.25), \mu_{v_c} = 2.5, \mu_{v_e} = 1.5, t_0 = 0.2$. L - 1e4 = Analytic likelihood with 10,000 trials per condition (purple); L = Analytic likelihood with 500 trials per condition (orange); PDA = PDA with 500 trials per condition (green). The lower panel shows one representative case. The upper panel shows 95% Bayesian credible intervals from 100 independent fits. The probability density of t_0 for L - 1e4 went up to almost 100. We zoomed in the y axis in the lower panel to [0, 35]. The credible intervals were ordered by 50% quantile.

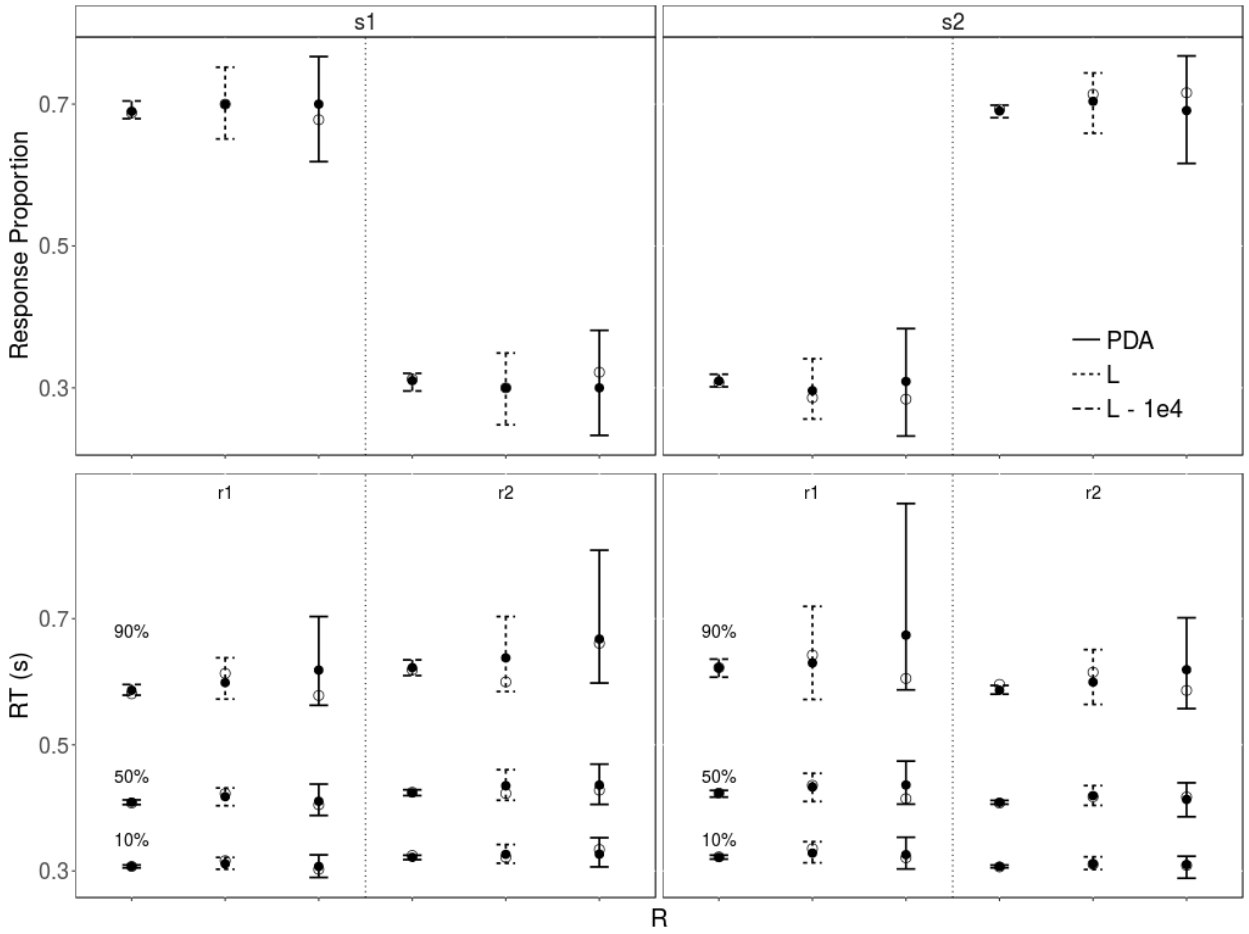

Figure 3. Goodness-of-fit. The figure shows one representative case selected from the 100 independent fits. The upper panel shows the response proportions (type: r1 & r2) to different stimuli (type: s1 & s2). The lower panel shows the response times for 10%, 50% and 90% percentiles. Likelihood calculation methods are represented by different line types. R = response type; PDA = PDA using 500 trials per condition; L = analytic likelihood using 500 trials per condition; L – 1e4 = analytic likelihood using 10,000 trials per condition. The white and dark circles represent data and model predictions, respectively. In the case of L – 1e4, because the models fit data closely, most circles overlap onto each other. The error bars showed 95% credible intervals.

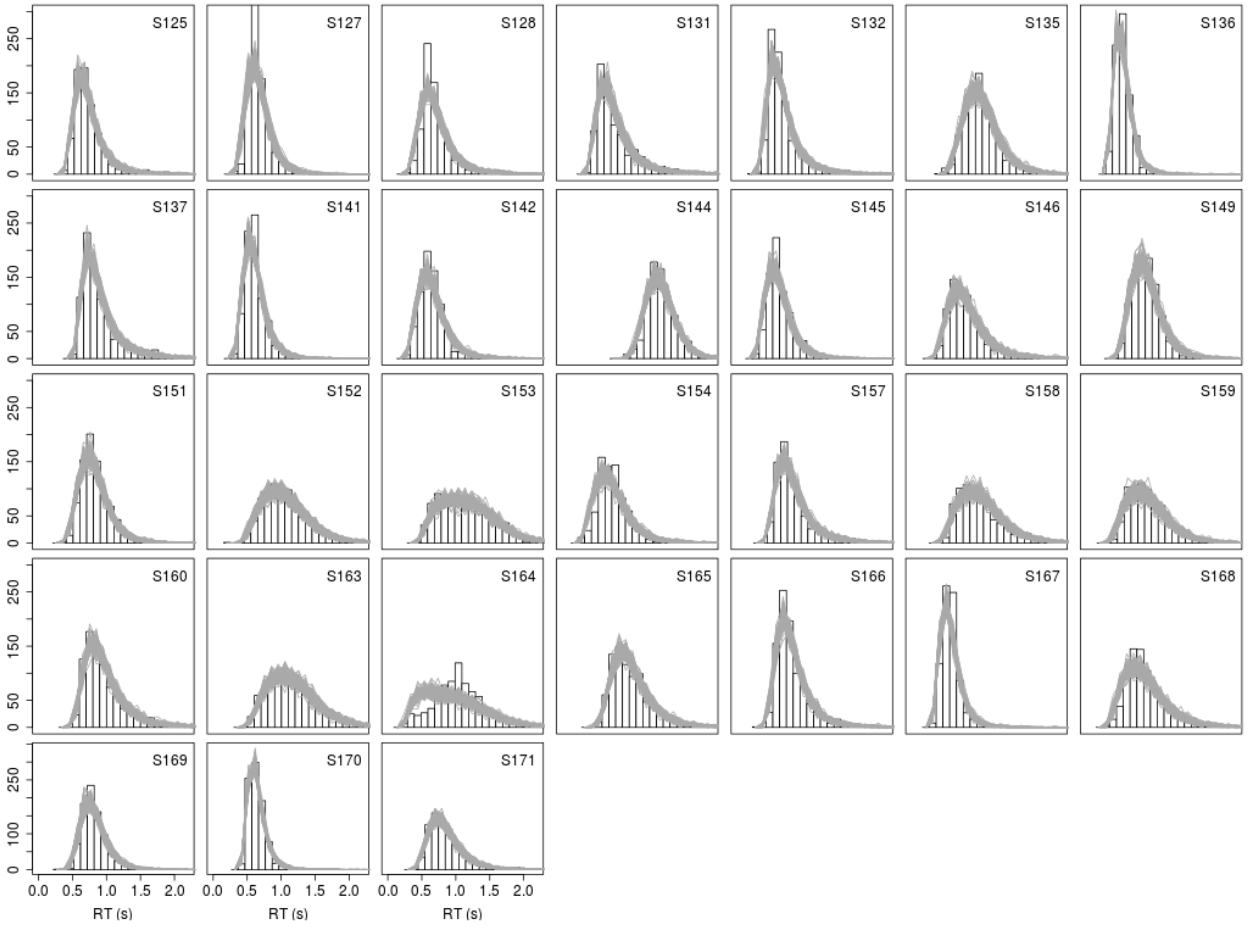

Figure 4. The fit quality of PLBA 1f model for the pre-switch correct responses. Gray lines represent the model predictions and the histograms are drawn based on the empirical data.

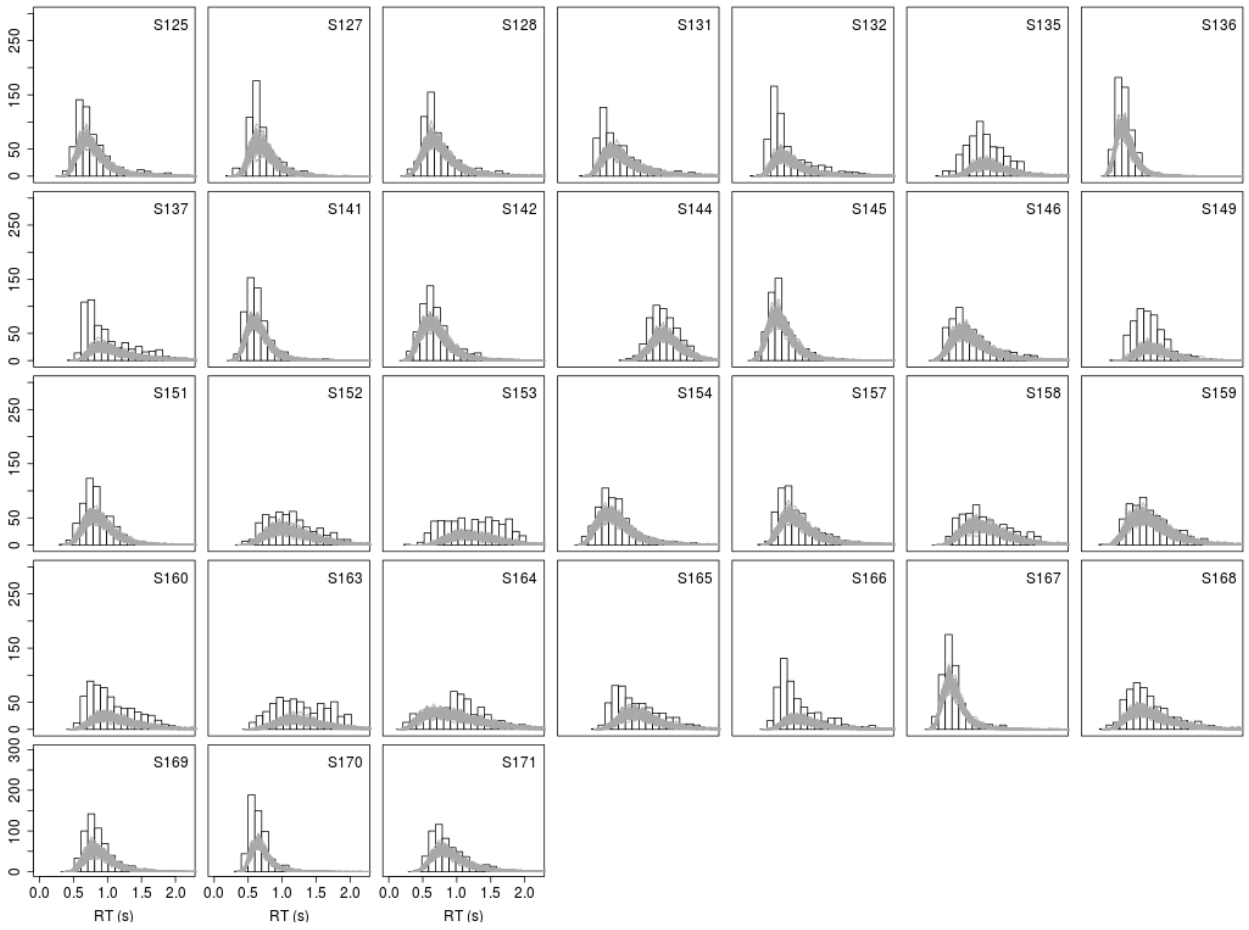

Figure 5. The fit quality of PLBA 1f model for the post-switch correct responses. Gray lines represent the model predictions. The histograms are drawn based on the empirical data.

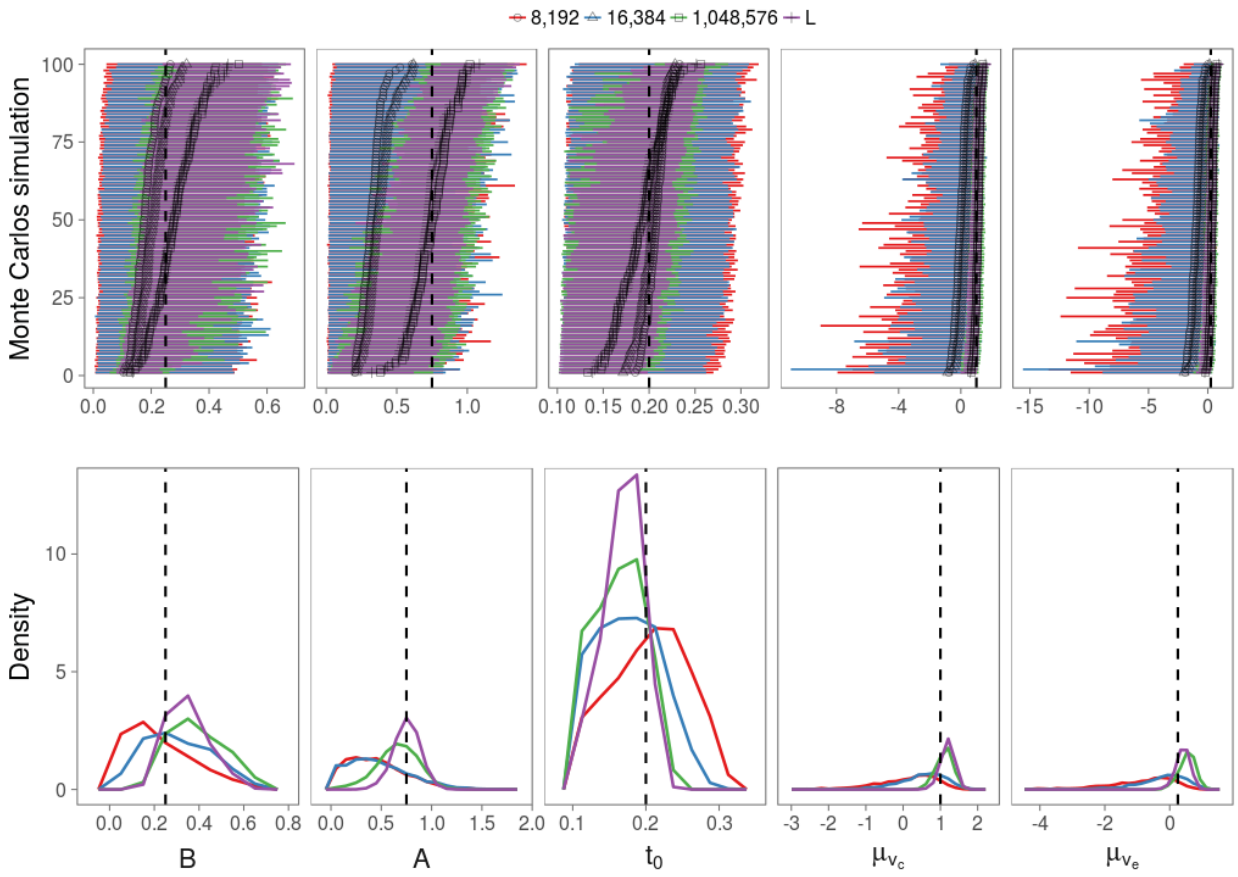

Figure 6. Marginal posterior distributions and Bayesian credible intervals (Study I). The lower panel shows one representative cases for each estimation method. The upper panel shows 95% Bayesian credible intervals from the 100 independent fits. The dashed lines show true parameter values: $b = 1.0, A = 0.75$ $(B = 0.25), \mu_{v_c} = 1.0, \mu_{v_e} = 0.25, t_0 = 0.2$. L = Estimated by analytic likelihood function with 500 trials per condition (purple); Green, blue, and red colors lines show PDA estimated distributions, using 1,048,576, 16,384, and 8,192 model simulations, respectively. All three PDA estimates were based on 500 trials per condition.

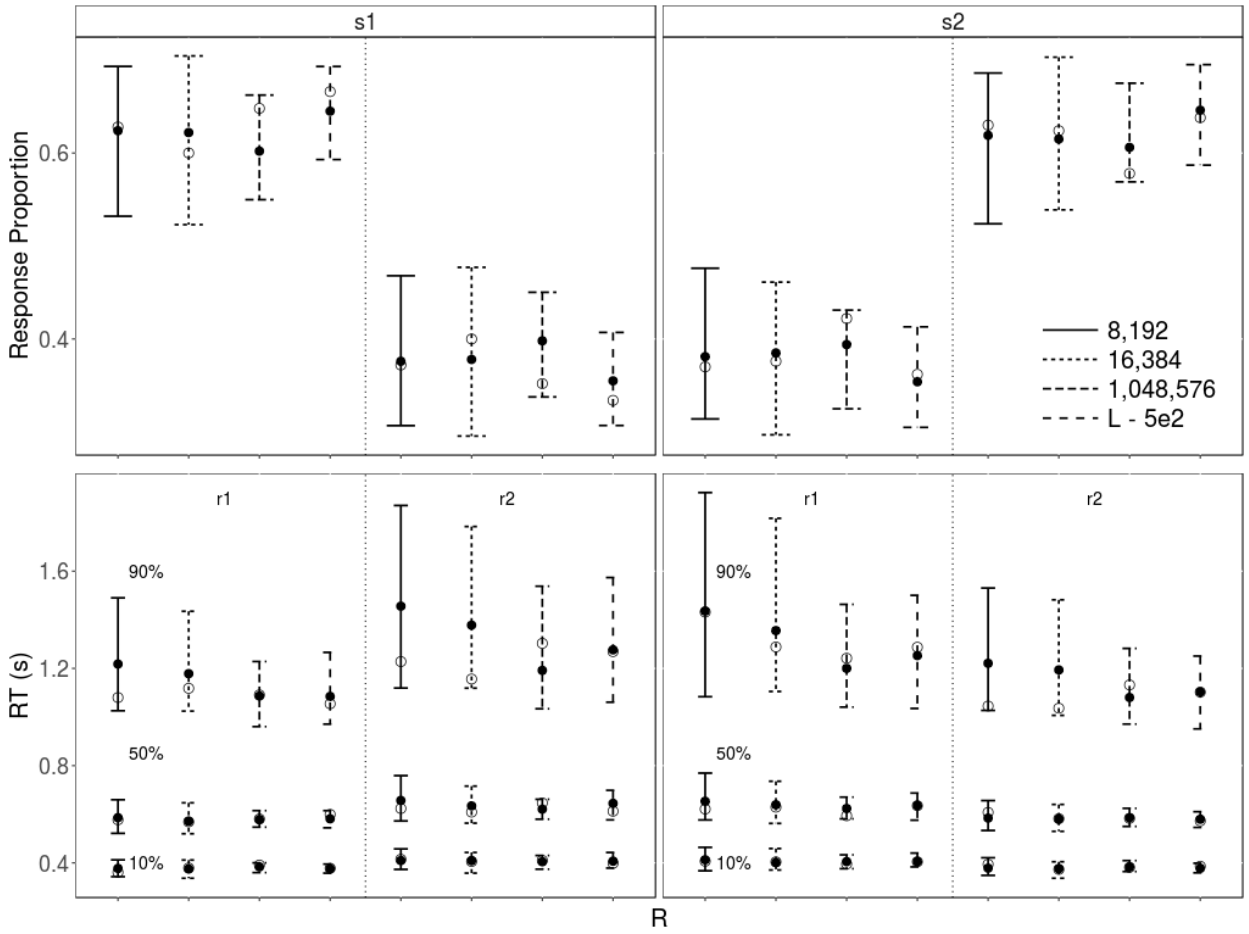

Figure 7. Goodness-of-fit (Study I). The figure shows one representative case selected from the 100 independent fits. The upper panel shows the response proportions (type: r1 & r2) to different stimuli (type: s1 & s2). The lower panel shows the response times for 10%, 50% and 90% percentiles. Likelihood calculation methods are represented by different line types. 8,192 = PDA approximation using 8,192 model simulations; 16,384 = PDA approximation using 16,384 model simulations; 1,048,576 = PDA approximation using 1,048,576 model simulations; L – 5e2 = analytic likelihood using 500 trials per condition. All likelihood calculation methods fit data with 500 trials per condition. The error bars showed 95% credible intervals.

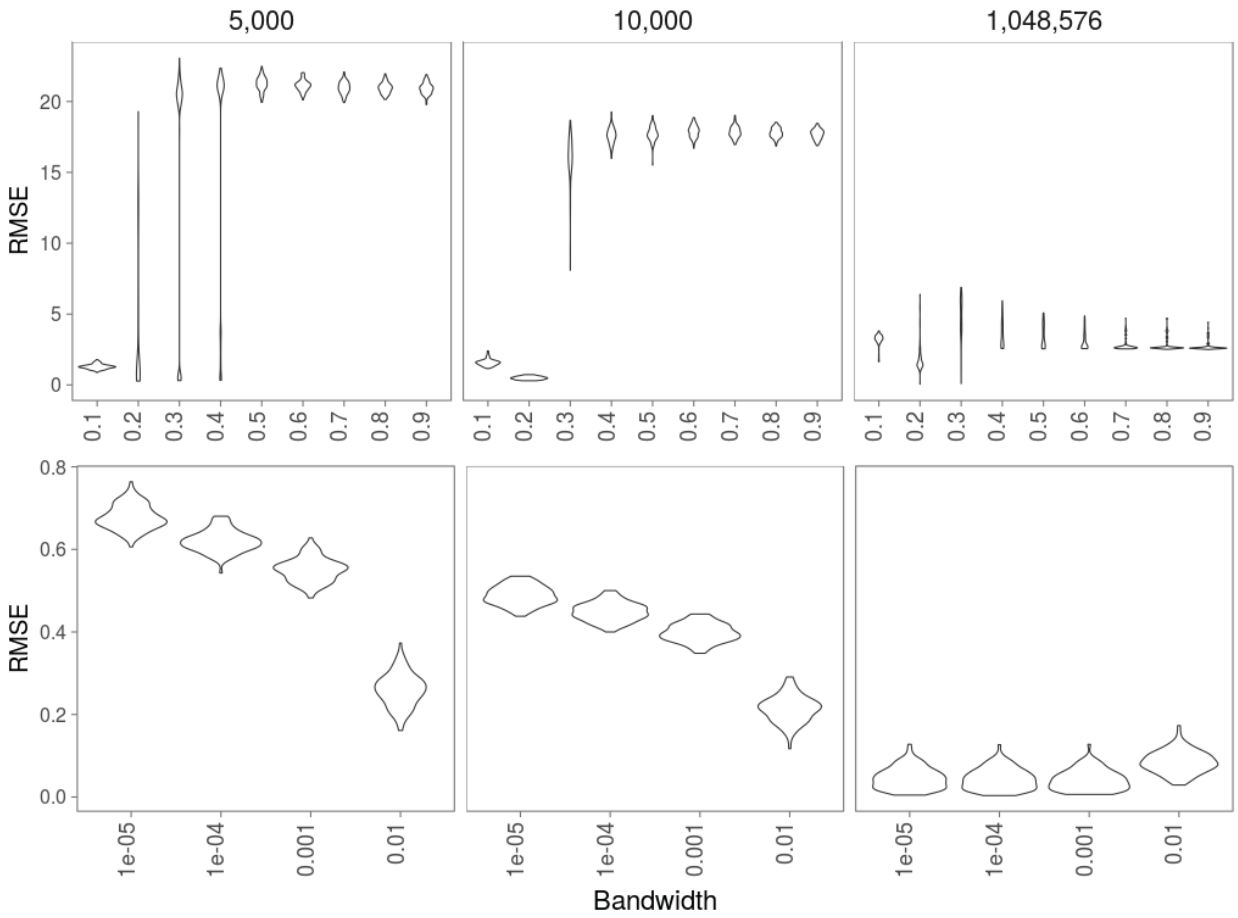

Figure 8. Bandwidth influences on biases. Each violin plot shows the distribution of root mean squared errors (RMSEs) across the parameters for 100 independent model fits. The upper ribbon indicates the numbers of model simulations.

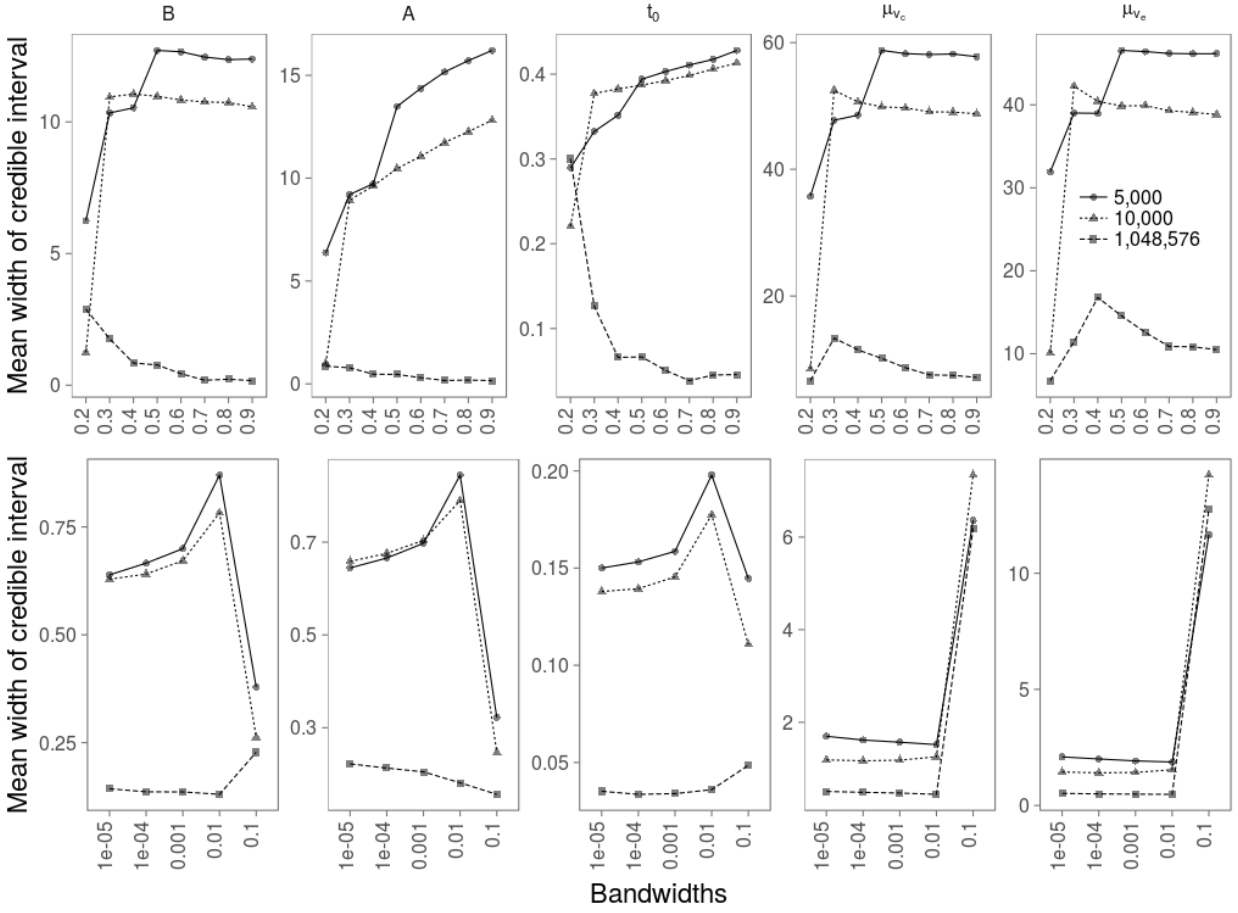

Figure 9. Bandwidth influences on variances. Each point represents the width of 95% credible intervals averaged, separately for each parameter, across 100 independent fits. The upper ribbon indicates different parameters. Different numbers of model simulations are represented by different line types and symbols.

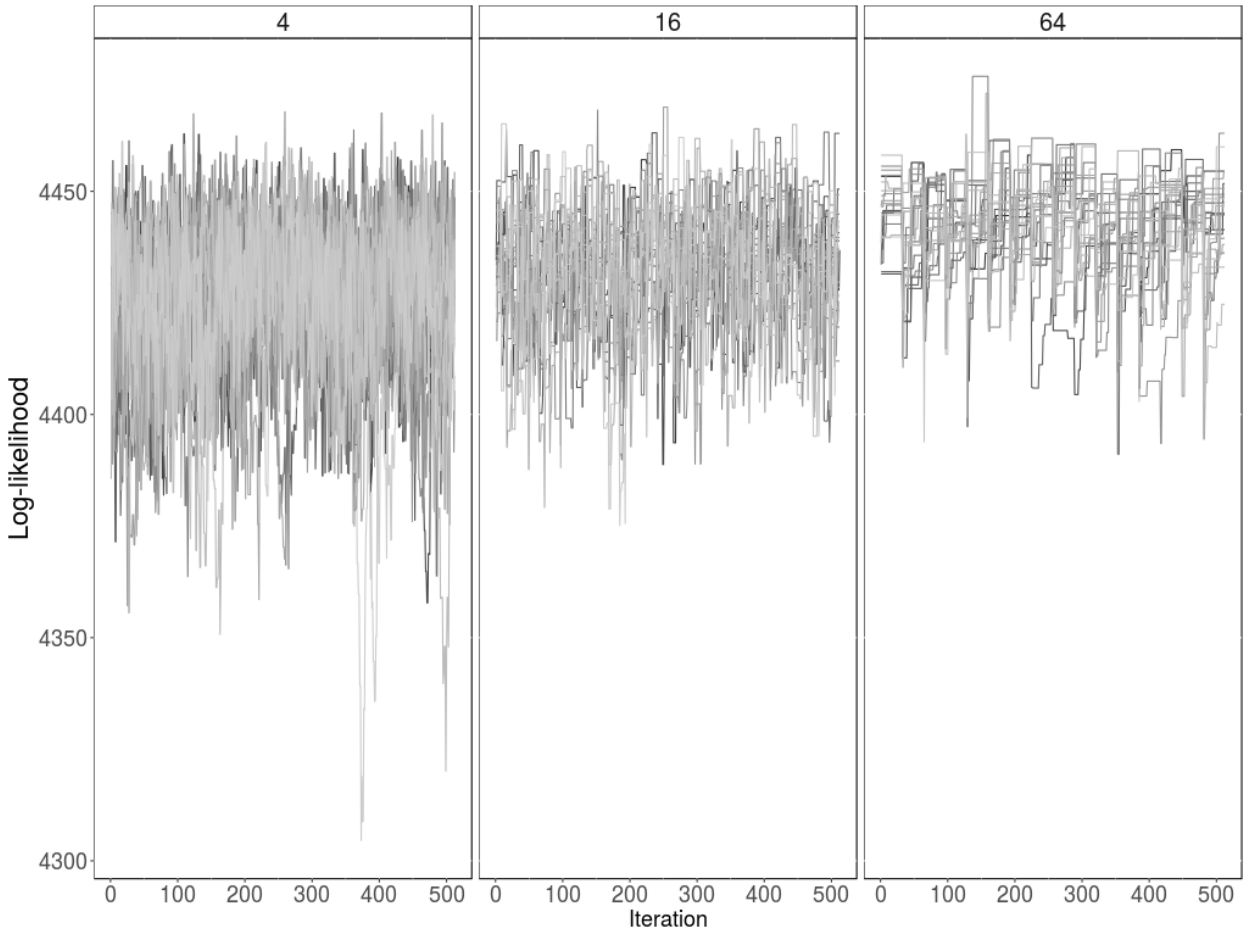

Figure 10. Trace plots show chain stagnation. The upper ribbon shows the recalculation intervals. The y axis shows posterior log-likelihoods.

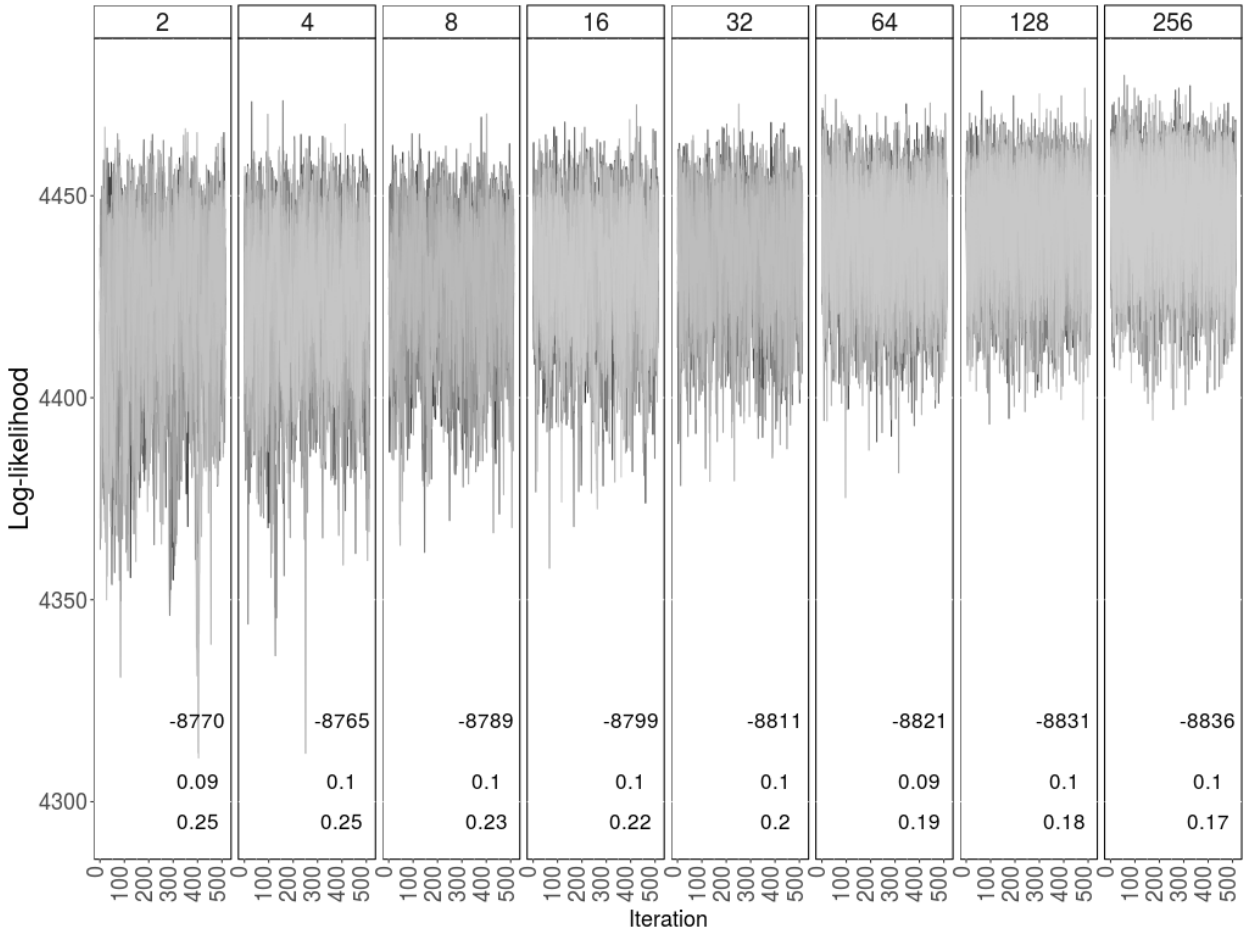

Figure 11. The Influence of recalculation intervals on model fits. The y axis shows posterior log-likelihoods. The upper ribbon shows different recalculation intervals. The texts inside each subplot show DIC, root mean squared errors (RMSEs), and the width of 95% credible intervals. The RMSEs and credible intervals are averaged across the parameters.

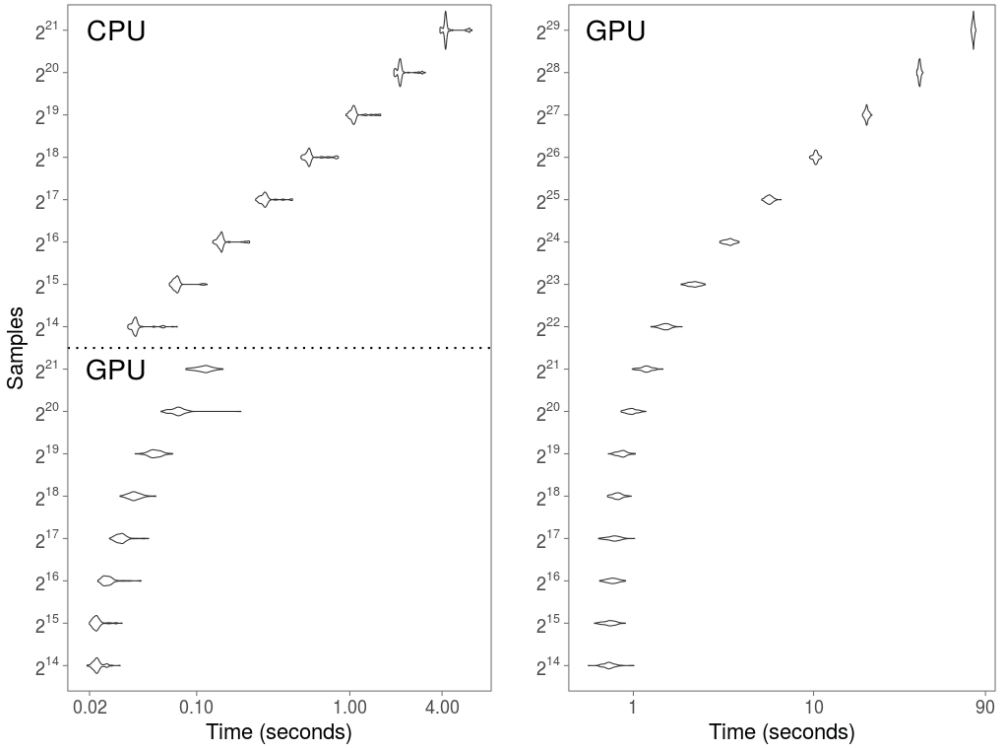

Figure 12. The violin plots of the computation times for calculating 10,000 PLBA densities. The y axis shows the numbers of model simulations for synthesizing a density function. The x axis is on a $\log_{10}$ scale. The violin plots show the distributions for 100 independent replications.

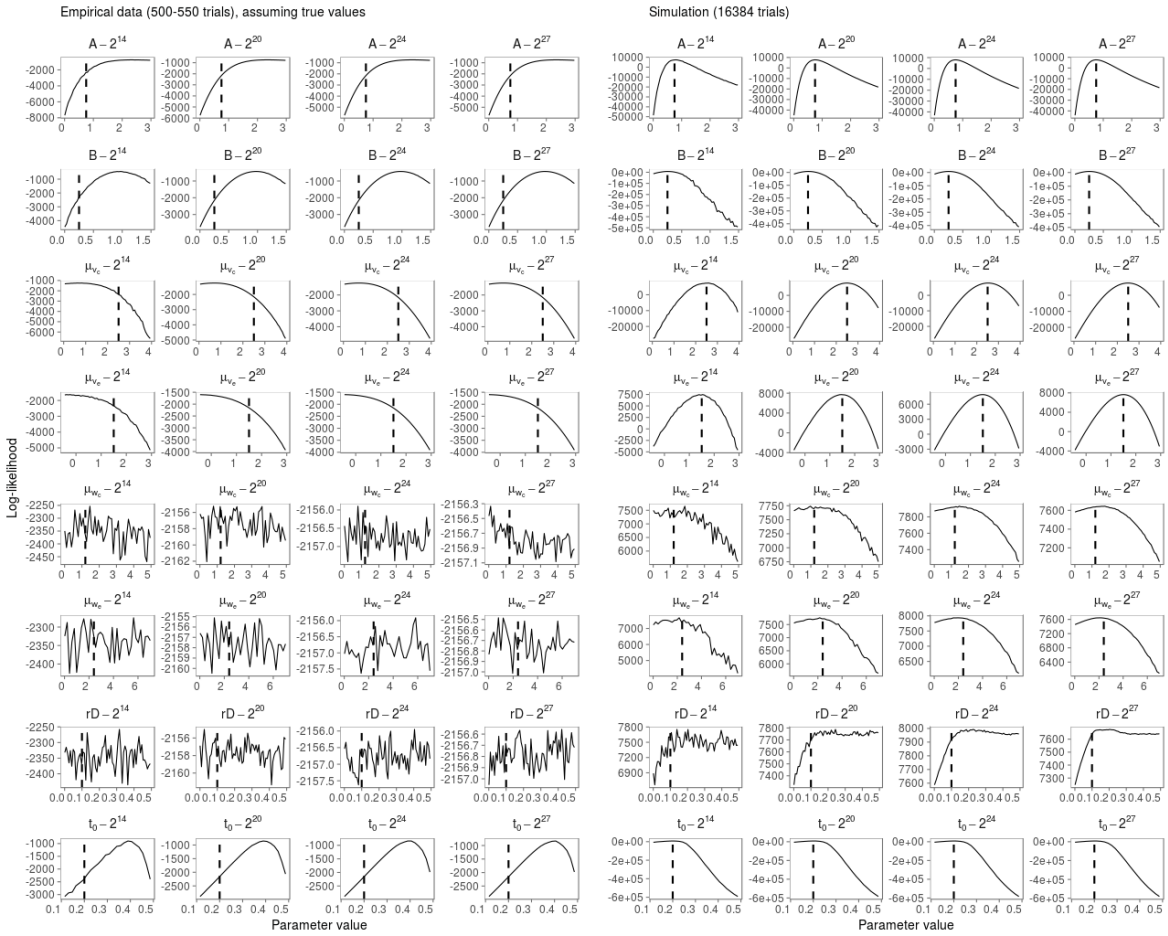

Figure 13. The likelihood profiles of the PLBA model. The profile plots examine if the numbers of model simulations and the data points are sufficient to recover parameters. We chose the parameters, $A = 0.75, B = 0.25, \mu_{v_1} = 2.5, \mu_{v_2} = 1.5, \mu_{w_1} = 1.2, \mu_{w_2} = 2.4, rD = 0.1, and\ t_0 = 0.2$ to profile the likelihood. The log-likelihoods in the left panel [Empirical data (500-550 trials), assuming true values] were calculated based on the empirical data (only the first participant). The right panel [Simulation (16,384 trials)] was based on the simulated data. The simulated data were drawn from the PLBA model, using an experimental design with one two-level stimulus factor, identical to the empirical data but fixed switch time at 0.5 seconds (Holmes et al. 2016 calibrated this time for each participant). From left to right and top to down, each row shows four different numbers of model simulations (16,384, 1,048,576, 16,777,216 vs. 134,217,728) for a PLBA parameter.

Appendix

PLBA *1f* model has eight parameters: the upper bound of the start point (A), the decision threshold (b), the mean drift rates of the choice one accumulator (μ_{v_1}) and the choice two accumulator (μ_{v_2}) in the first LBA process, the mean drift rate of the choice one accumulator (μ_{w_1}) and the choice two accumulator (μ_{w_2}) in the second LBA process, a common standard deviation of the drift rates for the two accumulators (σ), non-decision time (t_0), and a delay time (rD). Here we fixed σ at one, as a constant, so the model has eight parameters.

The PLBA model first draws the value of a start point from a uniform distribution with a range of 0 to A.

$$x_0 \sim U(0, A) \qquad (10)$$

At the first stage, the model draws the choice one and the choice two drift rates from two independent truncated normal distributions with means, respectively, μ_{v_1} and μ_{v_2}, and a common standard deviation, σ. After the sum of the switch time and the drift rate delay, the model draws two new drift rates from another two truncated normal distributions with mean drift rates, μ_{w_1} and μ_{w_1}.

$$v_1 \sim TN(\mu_{v_1}, \sigma), v_2 \sim TN(\mu_{v_2}, \sigma) \qquad (11)$$

$$w_1 \sim TN(\mu_{w_1}, \sigma), w_2 \sim TN(\mu_{w_2}, \sigma) \qquad (12)$$

v_1 and v_2 are the choice one and the choice two drift rates for a trial before the switch time and after the switch time, they are w_1 and w_2.

Before fitting the PLBA model to the empirical data we first conducted three mini-studies to check on the effect of approximation noise and whether the empirical data have sufficient trials to permit acceptable recovery of the PLBA parameters.

PLBA Simulations

Methods

The first mini-study calculated likelihoods for one of the participants in the empirical data with 537 trials (Holmes et al., 2016). The likelihood profile plot was constructed by fixing all parameters, except one, with switch time fixed at 0.5 second. For example, the upper left subplot in **Error! Reference source not found.** profiles the change in log-likelihood, with fixed $B = 0.25, \mu_{v_1} = 2.5, \mu_{v_2} = 1.5, \mu_{w_1} = 1.2, \mu_{w_2} = 2.4, rD = 0.1,$ and $t_0 = 0.2$, as *A* varies from 0.0075 to 3. The curve in this subplot hence shows the maximum log-likelihood of *A* happens when it is at 2.31. The second mini-study used the PLBA parameters, $A = 0.75, B = 0.25, \mu_{v_1} = 2.5, \mu_{v_2} = 1.5, \mu_{w_1} = 1.2, \mu_{w_2} = 2.4, rD = 0.1,$ and $t_0 = 0.2$ to simulate a with a very large number of trials (16,384 per condition). We selected these parameter values because they generate data sets like empirical data with responses that were influence by both the pre-switch and post-switch parameters. The first and the second mini-studies simulated, 2^{14} = 16,384, 2^{20} = 1,048,576, 2^{24} = 16,777,216, and 2^{27} =134,217,728 to synthesize PLBA probability density functions. We used these four cases to search for a minimal number of model simulation that can clearly locate maximum likelihoods for all PLBA parameters. The aim of these two studies is to examine the influence of the trial and model simulation numbers on estimating the PLBA likelihoods. Note in the first mini-study, we profiled the empirical data with an arbitrarily chosen parameters, so the true value lines in the left panel of Figure 13 [“Data (500-550 trials), assuming true values”] do not always match the maximum likelihood.

The third mini-study is a parameter recovery study. As commonly found in many evidence accumulation models (Turner et al., 2013; Holmes et al, 2016), correlated parameters can make parameter estimation difficult and imprecise when empirical studies provide

insufficient observations. Because the PLBA model does not have an analytic probability density function, our aim in the third study is to test whether PLBA parameters are identifiable in the limit of a large number of data points, 16,384 trials per condition, from the specific PLBA model with $A = 0.75, B = 0.25, \mu_{v_1} = 2.5, \mu_{v_2} = 1.5, \mu_{w_1} = 1.2,\ \mu_{w_2} = 2.4, rD = 0.1,$ and $t_0 = 0.2$. We then fit the PLBA model to the simulated data, using over ten million (2^{24}) model simulations, a case that can locate clear maximum likelihoods for the post-switch mean drift rates (Figure 13). That is, if we use 1,048,576 model simulations, PDA might return imprecise μ_{w_1} and μ_{w_2} values corresponding to maximum likelihoods. The very large numbers of trials and model simulations minimize the influence of data and approximation noise, so if we fail to estimate PLBA parameters precisely, we cannot hope to recover parameters in real data.

Results

The result of the first study indicates, when there are a similar number of data points to the empirical data, the likelihood maxima of the post-switch parameters are difficult to estimate, even with very large numbers of model simulations. The results of the second study show it is possible to locate clear maxima for those parameters with 16,384 data points and 2^{27} model simulations. Further, the result suggests that with 16,384 trials and 2^{20} model simulations, there might still be some difficulties in identifying *rD*. Although the result suggests an ideal number of model simulations to identify maximal likelihoods is 2^{27}, this huge number of model simulation will start introducing a new computational bottleneck. That is, when simulating more than 2^{24} samples (over 16 million), the GPU computation times return to a linear relationship with the numbers of model simulations (Figure 12). The two studies together indicate, it is unlikely the post-switch mean drift rates are identifiable with around 500 observations, regardless how many model simulations. Second, the approximation noise might hinder

recovering post-switch parameters with tens of thousands model simulations, which were often used in CPU-based PDA. The result of the third study suggests, in the asymptotic case (huge numbers of trials and model simulations), all mean drift rates and *A* parameters can be recovered to high precision, *rD* and t_0 estimates are biased toward upper end, and B estimate is based toward lower end (Table 2).

[Insert Figure 13 & Table 2 Here]

Footnotes