Parallel probability density approximation

Yi-Shin Lin¹ · Andrew Heathcote¹ · William R. Holmes²

Published online: 30 August 2019
© The Psychonomic Society, Inc. 2019

Abstract

Probability density approximation (PDA) is a nonparametric 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 in order to attain good precision. We introduce *Parallel PDA* (pPDA), a highly efficient implementation of this method utilizing the 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 conducted simulation studies to investigate various issues associated with 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

With the rapidly increasing capabilities of computer hardware and software in recent years, simulation-based approaches to investigating mathematical models of various phenomena have exploded. In the long history of using cognitive models to test theories, some of the work has been qualitative in nature, seeing whether a theory can predict the observed patterns of data. Other work has been quantitative in nature, determining whether a model/theory can quantitatively match features of the data. The latter approach typically involves fitting a model to data to determine how well the theory encoded in that model matches the observations. Most (though not all) past methods of achieving this have typically been limited to relatively simple models that 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, graphics processing unit (GPU)-enabled parallel implementation of canonical Bayesian Markov chain Monte Carlo (MCMC) methods that utilizes *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 method requires that the likelihood for the model of interest be analytically tractable (i.e., either solvable in terms of its fundamental functions or amenable to fast, stable, and accurate approximation through standard numerical integration methods), whereas 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 of 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

✉ Yi-Shin Lin
yishinlin001@gmail.com

¹ Division of Psychology, School of Medicine, University of Tasmania, Hobart, Tasmania, Australia

² Department of Physics and Astronomy & Department of Mathematics, Quantitative Systems Biology Center, Vanderbilt University, Nashville, TN, USA

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), which are calculated for both the data and the model simulations. This raises two central issues that are addressed by 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 nonparametric kernel density estimation (KDE; Silverman, 1986) to the simulated data in order to numerically approximate the model’s likelihood. The resulting summary both is sufficient and can be directly plugged into any procedure requiring likelihoods, such as Bayesian MCMC methods.

The PDA method was introduced into 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 the calculation of the KDE to the spectral domain. Holmes’s implementation also utilizes likelihood resampling to reduce the MCMC chain stagnation arising from likelihood approximation errors (see Holmes, 2015, for further discussion). In this article, we address the issue of the 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 theories by converting them to 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 (DDM; Ratcliff & McKoon, 2008), many do not (e.g., Cisek, Puskas, & El-Murr, 2009; Gureckis & Love, 2009; Thura, Beauregard-Racine, Fradet, & Cisek, 2012; Tsetsos, Usher, & McClelland, 2011). 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, because of the effort required to mathematically derive the complex formulas for these likelihoods. At worst, it may restrict the range of questions to which modeling can be 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 PDA, and of simulation-based methods more broadly, is twofold: first, to expand the scope of theories/models that can be tested, and 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 PDA is applied in a Bayesian modeling context, since these operations must be performed iteratively for multiple Markov chains. Furthermore, these operations introduce approximation errors that 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 the Compute Unified Device Architecture, CUDA C, a programming model accessing the power of GPUs. This heterogeneous programming model allows pPDA to harness the strength of both a computer’s central processing unit (CPU) and GPU to attain efficient and precise likelihood estimates. In a nutshell, the GPU conducts numerous (e.g., millions of) 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 being transferred back to the CPU for subsequent processing. Between these two steps, we optimize what is transferred to and from the GPU in order to minimize the data transfer burden that often plagues GPU-based computations. To ease the installation and use of pPDA, we have created an open source R (R Core Team, 2017) package, *ppda*, which is made available at our GitHub (<https://github.com/TasCL/ppda>). The user can easily access *ppda* using regular R installation methods.

In this article, 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-based studies in order to investigate the validity, scope, and best practices 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 article with a discussion of the specific issue of inflated likelihoods in PDA, of roadmaps for applying pPDA to other cognitive models, and of the future development of the ppda package. We have provided a detailed account of all analyses reported in this article 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 PDA and its application in Bayesian computation (see Turner & Sederberg, 2014, and Holmes, 2015, for further details, as well as the 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's rule:

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

Bayesian estimation can be carried out by MCMC methods without evaluating the denominator in Eq. 1, since the integral is often intractable, using:

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

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 parameters proposed (e.g., Hoffman & Gelman, 2014; Neal, 1994; Roberts & Rosenthal, 2001; Ter Braak, 2006; Turner & Sederberg, 2012). By combining these parameters with a data set, one can derive a number proportional to the posterior probability density by calculating the righthand side of Eq. 2. This proposed 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 proposed density to the reference density, so the denominator in Eq. 1 (which is not a function of the current parameters) is

irrelevant, as it cancels out. The ratio $\frac{\pi(\theta^*|y)}{\pi(\theta^{t-1}|y)}$ then guides an accept-or-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, but are otherwise rejected. The acceptance of less probable proposals sampled from low-density regions 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 likelihoods of the existing and proposed values of the 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 the 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, and 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 synthetic data set simulated from the model with parameters θ . From here on, $\tilde{\cdot}$ will indicate an approximate quantity. This renders the posterior probability density function

$$\pi(\theta|y) \approx \tilde{\pi}(y|x, \theta)\pi(\theta) \quad (3)$$

where this relation is now an approximate rather than exact proportionality, due to 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 step is of course model-dependent, but it is typically a simple, though computationally intensive, process. In the case of a response time model, this would require simulating many choice-and-response-time pairs. The second step is to construct an approximate log likelihood function on a discrete grid (e.g., a grid of time points) from those simulations using KDE (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

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

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

$$K_h(z) = \frac{1}{h} K\left(\frac{z}{h}\right) \quad (5)$$

where K integrates to 1 over the domain of interest. The kernel function could be any standard kernel. The specific mathematical trick used here to improve performance, 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

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

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 then transforming the result back. Hence, an efficient way to derive the approximated probability densities is

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

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

The detailed steps for 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 $\tilde{d}$ on a regular grid. (3) Fourier transform that discrete data function to produce $\mathcal{F}[\tilde{d}]$. (4) Multiply that transformed function by the transformed kernel $\mathcal{F}[K_h]$. (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 Step 1, the model simulation.

Cognitive models

Here we illustrate PDA in six well-known cognitive models in order 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, as well as diffusion decision (Ratcliff & McKoon, 2008) and linear ballistic accumulation (Brown & Heathcote, 2008) models. We choose these six models oft-used in the 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 for 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 process, 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).

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 the approximated densities directly from a Gaussian kernel, represented by a gray dotted line. This is the traditional KDE (Van Zandt, 2000). The second method uses the 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, compared 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, Wald, and Weibull models, shown in Figs. 1b, 1c, and 1d, respectively. The Wald model describes a continuous one-boundary diffusion process and can be used to account for RT data from simple detection experiments in which observers make only one type of response (Heathcote, 2004; Schwarz, 2002). The Weibull model

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

is used to describe RT data resulting from the asymptotic distribution of the minimum completion time of a large number of parallel racing processes. Again, the PDA estimations match the 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 that 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 the left side of the visual field. In this example, a data point consists of a pair of numbers: how much time an observer takes to respond (i.e., the RT) and which choice is made (i.e., right or 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 an RT 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 1.

The LBA accounts for choice as a race between independent, deterministic accumulators, in which 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. The RT is the sum of the time to make a choice and a nondecision time (consisting of the time to encode a stimulus and the time to produce a response). Three main types of parameters are 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 the 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 PDFs representing the RT distributions, to compare against the data. A similar process can be applied to the diffusion decision model (Fig. 1f), which assumes a single accumulation process with two boundaries, one for each response, 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 to trial.

The results in Fig. 1e and f show that using simulated data in conjunction with KDE produces RT distributions that closely match the analytic PDFs, with the only apparent differences being in the high-likelihood region around the mode of the distribution. As we will discuss later, this slight mismatch in high-curvature regions of the distributions are expected, since 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 apply 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 in order 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 the model parameters. A solution to 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 the likelihoods for each proposal with 30,000 model simulations for 5,000 MCMC iterations plus a 1,000-iteration 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 method is promising, an effective algorithm to deliver on 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 PDA

To resolve this dilemma for evidence accumulation models, pPDA transports the simulation parameters into GPU memory and then designates two memory pointers to the locations where the outcomes of the simulations will be stored. Two pointers are required because this method fits data with two dependent variables (i.e., choices and RTs). For 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 simulations 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 the information required for KDE back to CPU memory. These functions calculate the numbers of each choice, the maximum and minimum of the simulated RTs, and their standard deviation.² These statistics are then used to construct kernel bandwidths, the ranges of the bin edges, the bin edges, and the Gaussian filter. These operations are conducted in CPU memory. Next, pPDA transports the bin edge vector back to GPU memory in order to construct a histogram, which is then transported back to CPU memory. The histogram carries only 1,024 unsigned integers, so again pPDA operates well below the limitations of GPU bandwidth. More details about the pPDA implementation can be found on GitHub.

Replication study

Because pPDA is a novel method, we first examined whether it replicates the 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 Eq. 4. Also, the path width that can effectively accommodate the amount of information exchange between CPUs and GPUs is dubbed *bandwidth*, which is not the same as the kernel bandwidth in Eq. 4.

² We used 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 15 Markov chains, using the differential evolution MCMC (DE-MCMC; Turner et al., 2013) sampler, which is an adaptation of 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, pp. 383–384) by drawing a random number from a uniform distribution in every iteration. When the number was less than .05, the migration operator replaced the crossover operator in order to propose parameters. Even with such a long burn-in, some cases with small numbers of model simulations and large bandwidths had not yet converged, reaffirming the crucial roles of approximation precision and adequate bandwidth. Note that 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 bandwidth.

To get a sense of bias and variability, we conducted 100 independent fits for each case, in which 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 the start point, A , to 0.75; the drift rate for correct responses, μ_{v_c} , to 2.5; the drift rate for error responses, μ_{v_e} , to 1.5; and the nondecision time, t_0 , to 0.2. Following their parameter setup, 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, as compared to the number used previously (Turner & Sederberg, 2014). The number of model simulations must be a power of 2 because we used two specific parallel programming techniques, parallel reduction and for-loop unrolling in CUDA code (Harris, 2007), to speed the GPU computation.

Results

Figure 2 shows the results of the posterior parameter distributions estimated by pPDA (green) as compared to those

estimated by the analytic approaches (orange and purple). The marginal distributions for each parameter were plotted together on one canvas, using the same bin widths. This shows that, as would be 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 produced less accurate and precise parameter estimations than the analytic estimates using the same data sample size. Although they showed much larger variability than their analytic counterpart, the PDA distributions covered the true parameter values, as is shown in their 95% credible interval (upper panels in Fig. 2) and are comparable to those reported by Turner and Sederberg (2014; see their Fig. 3 upper panel).

Figure 3 shows that 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 the new data and by plotting the new data with the original data. The 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 the MCMC posterior likelihoods (Spiegelhalter, Best, Carlin, & Van Der Linde, 2002). On average, the DIC (−294.63) was substantially larger for pPDA than for the analytic likelihood function (−425.16). This occurred (1) because the complexity penalty equalled the variance of the posterior deviance (where the deviance was −2 times the logarithm of the likelihood)³ and (2) because Monte Carlo error increases the variability of the posterior likelihood estimates, inflating the DIC. This illustrates why Holmes (2015, p. 19) cautioned against comparing the DIC for a model fit by PDA with the DIC for models fit using analytic likelihoods. The same issue applies in the comparison of DICs between PDA methods that inflate posterior likelihood variability to different degrees (e.g., by using different numbers of model simulations).

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

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 this estimate measures the distance between the middle of the posterior deviance distribution (i.e., its mean) and its leading edge (because the deviance of the mean parameters is an estimate of the minimum deviance).

Fig. 2 Marginal posterior distributions and 95% Bayesian credible intervals. The dashed lines show the 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 panels show one representative case. The upper panels show 95% Bayesian credible intervals from 100 independent fits. The probability density of t_0 for $L - 1e4$ went up to almost 100. We zoom in the y-axes in the lower panels to [0, 35]. The credible intervals were ordered by 50% quantile

Fitting the 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), in which coherent motion in one direction switches to a 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 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 delayed change when the stimulus change affected 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 the process after the switch, possibly at a delay. Specifically, we fit Holmes

et al.'s (2016) 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 that only the drift rate is affected. Instead of fitting the hierarchical PLBA model as was done by Holmes et al., we fit the model to each participant's data separately. In the Appendix, we present a thorough examination of the PLBA model with three other simulation studies.

Method

We fit both the PLBA Model 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 3 and a standard deviation of 1. For rD and t_0 , we used uniform distributions with bounds of 0 and 1.

We deployed 24 chains and retained one sample every eighth iteration. During the first 100 iterations, we used

Fig. 3 Goodness of fit. The figure shows one representative case selected from the 100 independent fits. The upper panels show the response proportions (types: r1 & r2) to different stimuli (types: s1 & s2). The lower panels show the response times for the 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 the data and the model predictions, respectively. In the case of L-1e4, because the models fit the data closely, most circles overlap each other. The error bars showed 95% credible intervals

crossover and *migration* (5%) operators, and then only the *crossover* operator for the rest of the iterations. We kept running the iterations until all chains were well-mixed and stationary and had at least 1,024 effective samples. For both the PLBA and LBA fits, we used 1,048,576 model simulations and recalculated the likelihoods every fourth iteration. To optimize speed, we used the R package *ggdmc* (<https://github.com/yxlin/ggdmc>), which implements the DE-MCMC sampler (Turner et al., 2013) in C++. Furthermore, 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. The three groups of participants were fit in parallel in three identical virtual machines. Each machine was equipped with one Nvidia K80 GPU and one 12-core Intel CPU. Then we ran a pseudo-parallel scheme in each virtual machine, allowing multiple CPU cores seemingly to run in parallel, but in fact each CPU core interacted 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 used the DIC to assess whether PLBA Model 1f accounted for the data better than did the LBA model. Note that it is important for this comparison that the same PDA methods (i.e., 1,048,576 model simulations with a kernel bandwidth of 0.01 s) be used for both models because, as we showed earlier, changes in method can change the absolute DIC. We chose 2^{20} model simulations because the results of the PLBA simulation studies suggested that it is possible to identify some postswitch parameters with this setting, and it is relatively more efficient than using 2^{27} model simulations. We calculated the DIC separately for each participant and summed over them in order to assess the overall fits.

Table 1 shows that although most participants showed a smaller DIC in the PLBA than in the LBA fits (20/31), only two participants reached differences larger than 10. Seven participants showed no difference, and four had a larger DIC in the PLBA than in the LBA fit. When aggregated across the participants, the total DIC for the LBA and PLBA 1f models were, respectively, 31,571 and 31,530, with a difference of 41

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

	S125	S127	S128	S131	S132	S135	S136	S137	S141	S142
LBA	861	325	1,096	1,286	664	947	– 323	987	224	923
PLBA	860	326	1,095	1,288	663	947	– 324	991	222	921
	S144	S145	S146	S149	S151	S152	S153	S154	S157	S158
LBA	821	749	1,512	504	864	1,766	1,829	1,335	1,279	1,790
PLBA	812	747	1,511	500	861	1,766	1,815	1,333	1,278	1,801
	S159	S160	S163	S164	S165	S166	S167	S168	S169	S170
LBA	1,752	1,253	1,670	2,358	1,358	575	166	1,628	566	– 422
PLBA	1,751	1,254	1,665	2,351	1,357	573	166	1,628	567	– 421
	S171									
LBA	1,229									
PLBA	1,227									

S stands for subject/participant.

favoring the PLBA model. The hierarchical model fits reported by Holmes et al. (2016) found DICs for the LBA and PLBA models of 1,007 versus 926, a larger advantage of 81 favoring PLBA Model 1f.

Figures 4 and 5 show the quality of fits of the RT distributions for the preswitch and postswitch correct choices, separately for each participant. These figures were produced by first constructing the data histograms. We then randomly

sampled 100 sets of PLBA parameters without replacement from the posterior distribution in order to simulate 100 posterior predictive data sets. Each set is represented by one gray line in the figures, resulting in gray ribbons. In general, PLBA Model 1f fit the preswitch correct responses closely, whereas when a correct response was defined by matching the stimulus direction after a switch, the model fit the data less closely.

Fig. 4 Fit quality of PLBA Model 1f for the preswitch correct responses. Gray lines represent the model predictions and the histograms are drawn on the basis of the empirical data

Fig. 5 Fit quality of PLBA Model 1f for the postswitch correct responses. Gray lines represent the model predictions. The histograms are drawn on the basis of the empirical data

Best practice of the PDA method

We conducted three LBA simulation studies to examine the effectiveness and accuracy of the pPDA in recovering the data-generating parameters (i.e., parameter recovery studies; see Heathcote, Brown, & Wagenmakers, 2015) and to provide guidelines for applying PDA. To foreshadow, these studies suggest, first, that one must take noise introduced by PDA into account in making inferences based on PDA model fits. For example, the model selection and parameter uncertainty calculated from PDA fits differ quantitatively from those calculated using analytic likelihood functions. Second, whenever possible, one should conduct model simulations at the scale of millions in order to construct simulated PDFs. Third, in the case of fitting RT data, one should use a kernel bandwidth of approximately 0.01 s, which provides a good balance between estimation bias and variance. Finally, to avoid profound decrements in MCMC efficiency, one must modify the standard practice of reusing the 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 0. The prior distribution for nondesideration time was uniform, with a lower bound of either 0.1 or 0.01 s and an upper bound of 1 s.

Study I

The benefit of scaling up model simulations toward the asymptotic 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 100 independent fits. Figure 5(a) in Holmes’s study also demonstrated a case in which the simulated PDF became very close to the 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 investigated this case. We ran 100 independent fits and systematically compared the results using one million model simulations with the results based on lower numbers of model simulations.

Method For this study, we set μ_{v_c} and μ_{v_e} to 1.0 and 0.25, respectively, and used the same A , b , and t_0 as in the replication study. This set of parameters resulted in data with around 63% accuracy and generated slower errors than the parameters in the replication study, whose accuracy was about 70%. We conducted three parameter recovery studies, each with a different number of model simulations. The first used 8,192, which was less than the number used in Turner and Sederberg (2014), so we expected worse parameter recovery. The second and third used 16,384 and 1,048,576 simulations, respectively. We expected that the third study would show asymptotically unbiased and consistent estimations, as has been 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 number of model simulations on parameter estimation, highlighting three findings. First, as in the replication study, PDA introduced additional estimation error, which was markedly greater for 8,192 and 16,384 (red and blue lines) than for 1,048,576

simulations. Second, an increase in the simulation numbers decreases the approximation noise. Third, when the number of model simulations is 1,048,576, the PDA posterior distributions are almost the same as the analytic posterior distributions for the parameters μ_{v_c} and μ_{v_e} , and are very close for the parameters B , A , and t_0 . Figure 7 shows that the PDA method has fits almost identical to the analytic likelihoods, in terms of both response proportions and RT quantiles, when the simulations are above one million. The DIC values are 1,820, 1,772, and 1,621 for the three PDA estimates with increasing numbers of model simulations, and 1,556 for the analytic likelihood estimate. Hence, even with over one million model simulations, the inflation in DICs is sufficient to mandate against comparison with the DIC based on analytic likelihoods, since a DIC difference of 10 is considered large.

Study II

As was noted by Van Zandt (2000), the number of model simulations exerts its influence through the kernel function, which is modulated by the kernel bandwidth. In this study we investigated the role of the kernel bandwidth in association

Fig. 6 Marginal posterior distributions and Bayesian credible intervals (Study I). The lower panels show one representative case for each estimation method. The upper panels show 95% Bayesian credible intervals from the 100 independent fits. The dashed lines show the 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 indicates values estimated by the analytic likelihood function with 500 trials per condition (purple). The green, blue, and red 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

Fig. 7 Goodness of fit (Study I). The figure shows one representative case selected from the 100 independent fits. The upper panels show the response proportions (type: r1, r2) to different stimuli (type: s1, s2). The lower panels show response times for the 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 the data with 500 trials per condition. The error bars show 95% credible intervals

with the number of model simulations. A variety of methods have been proposed for finding an optimal bandwidth for KDE (Chiu, 1991; Goldenshluger & Lepski, 2011; Silverman, 1986). Silverman's (1986) plug-in method is perhaps one of the most widely used and intuitive methods, but it may not always be optimal. We investigated the optimal bandwidths for RT data for relatively simple decisions, spanning the range from 0.1 s 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, pp. 38–39), provide general guidance for the selection of testing bandwidths.

$$\text{Bias}(\hat{f}(y)) \approx \frac{h^2}{2} f''(y) M_2(K) \quad (8)$$

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

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 , $\|K\|_2$ denotes its Euclidean distance (i.e., the L^2 norm), and 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 the bias and variance of density estimates. Selecting a small bandwidth reduces bias, but as is 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 to choose a number of simulations large enough to reduce the variance sufficiently. The latter condition is greatly facilitated by use of a GPU, since it makes possible the use of large N_s .

Method We examined the bandwidth selection problem by testing 13 different potential bandwidths for choice RT data (on the scale of seconds): 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 executed using only the CPU, because when the number of model simulations is less than 2^{15} ($= 32,768$), CPUs tends to outperform GPUs. Each of the 39 combinations was conducted independently 100 times each. We expected that the influence of bandwidth on variance should become negligible when the number of model simulations was very large, and that the influence should become apparent when the

number of model simulation was small. In this study we used pPDA to fit data sets with 500 trials per condition.

The reason that we tested absolute bandwidths instead of using, for example, Silverman's (1986) method was 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 then it often suggests a very large bandwidth, which is clearly suboptimal. Although in Silverman's method one can choose adjusted interquartile ranges instead of standard deviations as its bandwidth suggestion, this method does not mitigate the problem, because when the number of model simulations is very large, the slow RTs become very slow. Instead, given that we know the typical scale of our data from the previous literature, we selected a wide range of possible bandwidths that might yield optimal PDA performance in fitting the RT data.

Results Figure 8 presents the distribution over 100 replications of the root mean squared estimation errors (RMSEs) averaged over parameters, and Fig. 9 shows the widths of the 95% credible intervals for each parameter. When the bandwidth is greater than 0.1 s, both the biases (Fig. 8) and variances (Fig. 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 the bandwidth is less than 0.01 s, but they 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 s for 5,000 and 10,000 model simulations, whereas the minimum for μ_{v_c} and μ_{v_e} is at 0.01 s. For over one million model simulations, variance is fairly constant for 0.1 s or less, except for μ_{v_c} and μ_{v_e} where it is higher for 0.1 s and low and constant for smaller bandwidths.

In summary, this study suggests that a bandwidth between 0.01 to 0.1 s 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 it imposes an additional computational burden by requiring the likelihood of accepted samples sometimes to be recalculated, prevents the problem of chain stagnation. It remains an unexplored question, how different recalculation intervals would affect the performance of Bayesian MCMC. In this study we examined this question. We recalculated the likelihoods of accepted samples every 2, 4, 8, 16, 32, 64, 128, and 256 consecutive iterations.

Fig. 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 labels indicate the numbers of model simulations

Fig. 9 Bandwidth influences on variances. Each point represents the width of the 95% credible intervals, averaged separately for each parameter, across 100 independent fits. The upper labels indicate the

different parameters. Different numbers of model simulations are represented by different line types and symbols

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

Results We deployed 30 chains, discarded the first 512 samples, and kept every second sample for the next 512 iterations so that the way the chains moved would be evident. The results are plotted in Fig. 10 in terms of sampled posterior log-likelihood values for each chain. Figure 10 presents only three recalculation intervals—4, 16, and 64—with longer step-like intervals showing increasing chain stagnation (i.e., periods of constant likelihood) as recalculation becomes more infrequent. An initial finding suggested that chain stagnation is a problem with regard to efficiency, rather than one that prevents chains from converging.

We then ran further iterations, adjusted the thinning intervals, and stopped the model fits until the chains became

stationary, well-mixed, and had at least 512 “effective” samples (i.e., once adjusted for the effects of autocorrelation). Figure 11 shows an increasing bias toward 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 the DIC to decrease, with the higher average indicating a spuriously better fit, and reduced variance indicating a spurious decrease in model complexity. For the intervals of 2 and 4, the results appear to be reasonably stable.

Discussion

Taken together, the simulation studies provide guidelines for applying PDA when fitting RT data. First, 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. As compared to using the analytic likelihood, PDA with over one million model simulations gives similarly good estimations for drift rates, but it results in some differences in the other parameters.

Fig. 10 Trace plots showing chain stagnation. The upper labels show the recalculation intervals, and the y-axis shows posterior log-likelihoods

Third, an optimal bandwidth in PDA for fitting RT data is around 0.01 s. Fourth, it is best to recalculate as frequently as possible, given sufficient computational resources, and at least every fourth iteration, as was 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, or between PDA results with different numbers of simulations or different recalculation intervals.

Performance profiling

We conducted three performance profiles to compare pPDA with conventional, CPU-based PDA. First, we measured the time the two methods take to fit the LBA model to simulated data. pPDA addresses two computational bottlenecks: those resulting from (i) drawing many model simulations and (ii) synthesizing these simulation samples into a likelihood (Holmes, 2015, p. 22). The first performance profile was

Fig. 11 The influence of recalculation intervals on the model fits. The y-axis shows posterior log-likelihoods, and the upper labels show different recalculation intervals. The text inside each subplot shows, in descending

order, the DIC, root mean squared errors (RMSEs), and width of the 95% credible interval. The RMSEs and credible intervals are averaged across the parameters

conducted to reveal the difference these two improvements make, and in the second we compared the influences of the number of model simulations on the computation times of pPDA and of CPU-based PDA. In the second performance profile, we simulated 10,000 trials based on the PLBA model with the same parameters as in the third PLBA simulation study above (see the [Appendix](#)), calculating 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 100 times. The third performance profile, similar to the second one, measured the time required to calculate 10,000 PLBA probability densities, but only for the 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 profile was to determine the extent to which allocating many GPU memories itself becomes a computational bottleneck.

To make a fair comparison, the regular PDA was done by recoding the MATLAB PDA method from Holmes (2015) in C++ and including it our R package, *ggdmc*. Using identical methods of software packaging would allow us to compare the two methods in similar computational environments, so any 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 capacity for computing 12 processes in parallel. This desktop computer was equipped with an Nvidia Tesla K20 GPU and a GeForce 980 GPU. Note that because we coded pPDA in CUDA C, it works only with Nvidia GPUs. Second, we timed the CPU-based PDA on four identical virtual machines. Each machine was configured with an Intel 12-core CPU running at a 2.6-GHz clock rate and an Nvidia Tesla K80 GPU. We conducted 200 Bayesian fits on the empirical data (Holmes et al., 2016), using a five-parameter LBA model (comprising start-point variability, decision threshold, nondecision 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 the CPU core. Hence, both methods drew on 1,048,576 model simulations to synthesize PDFs. All model fits ran 512 iterations using a mixture of the *crossover* (Ter Braak, 2006) and *migration* (Turner et al., 2013) operators, and then ran another 1,024 iterations using only the *crossover* operator. Both methods recalculated the 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 the GPU and CPU.

Results

As is shown in Table 2, on average, pPDA finished one Bayesian model fit in 43 h, which was almost three times faster than the CPU-based PDA [120 h; $t(99) = 40.34$, $p < .0001$]. Table 2 shows that the fitting times of the CPU-based fits ($SD = 19.25$ h) were also more variable than those for the pPDA fits ($SD = 0.34$ h). The maximum time for CPU-based PDA to finish a fit was 143 h, but sometimes it finished in only 84 h, whereas all of the GPU computations were completed in around 42 or 43 h. This was 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 (in terms of, e.g., CPU and GPU temperatures, available RAM, etc.), but the latter drew each model simulation in a slightly different setting. This suggests that with respect to computation time, pPDA is also more predictable.

Figure 12 shows the times for calculating PLBA probability densities. There are four key findings here. First, the computation time of the CPU increases linearly (on a $\log_{10}$ scale) with the number of model simulations. Second, the time it takes for a CPU to calculate 10,000 PLBA densities using 16,384 model simulations (median = 40 ms) is similar to that of a GPU using 262,144 model simulations (median = 39 ms), a speedup by a factor of 16. Third, GPUs scale better than CPUs as the number of model simulations increases. It takes more than 4 s (median = 4,220 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, improvement by a factor of 37. Finally, when the number of model simulations is up to 2^{23} (over eight million), the burden of handling large GPU memory spaces gradually manifests. When the number of model simulations is more than 2^{25} , the computation time of GPUs also becomes linear (on a $\log_{10}$ scale) with the number of model simulations.

Table 2 Performance profiles for CPU-based PDA and pPDA (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

Fig. 12 Violin plots of computation times for calculating 10,000 PLBA densities. The y-axis shows the number of model simulations for synthesizing a density function; note that the x-axes are on a $\log_{10}$ scale. The violin plots show the distributions for 100 independent replications

General discussion

Parallel PDA makes approximate Bayesian computation practical for fitting cognitive models for which the likelihood functions are often intractable. PDA has its roots in KDE (Parzen, 1962; Silverman, 1986; Van Zandt, 2000), a method for estimating probability densities from Monte Carlo simulated data. By harnessing heterogeneous GPU/CUDA and CPU/C++ programming models, 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 article 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 the use of CUDA limits this approach to Nvidia hardware, it allows the user an easy path to harnessing massively parallel GPU computation via the accessible R language. The challenge others have encountered (e.g., Verdonck et al., 2016) is to transfer synthesized data occupying a huge amount of GPU memory back to the CPU side for handling. Instead of choosing the strategy of fine-tuning the CUDA stream scheduling, we opted 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 has been documented by 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 of the synthesized data. Therefore, profiling memory usage and its impact on performance is simply not applicable in our case, because our pPDA transfers very tiny amounts of memory in and out of the GPU. This tool 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 that one should use GPUs to synthesize simulated histogram with as many model simulations as possible, such as over one million; set a bandwidth at least smaller than 0.01 s, and perform the modification of standard Metropolis methods suggested by Holmes (2015), by 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 to other complex models (Holmes & Trueblood, 2018; Miletić, Turner, Forstmann, & van Maanen, 2017; Trueblood et al., 2018), demonstrates that one can apply pPDA to fitting intractable cognitive models. In the following sections, we discuss the problems for MCMC methods caused by PDA, the limitations of our approach, and its future development.

Sampling problems caused by PDA

One problem in applying PDA to Bayesian computation is *likelihood inflation* (Holmes, 2015). This problem causes spuriously large likelihoods, due to noise in the PDA estimates. Likelihood inflation results in chain stagnation, as other plausible proposals are rejected in favor of spuriously likely samples, so that the chain remains unchanging. Fortunately, this problem can be resolved by recalculating the likelihoods of accepted samples, but this would double the computational cost if it were done on every occasion. We investigated mitigating this extra cost by performing recalculation less often. We found that long recalculation intervals, although they slow down chain mixing, do not prevent Markov chains from reaching convergence. However, although they save computation, such intervals spuriously reduce variability, and so can produce an overly optimistic picture of the level of estimates' certainty.

Limitation and future development

PDA, although it applies to a wide range of cognitive models, does not solve all modeling obstacles, and it introduces some pitfalls, with computation time and approximation noise being the most prominent. The purpose of this article has been both to 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 influence of future GPUs and CUDA libraries on ppda. Here we provide recommendations based only on the four types of GPUs 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 Compute Capability versioning system (Nvidia, 2018, p. 15). For example, we found that the Tesla K80, with a 3.7 Compute Capability, calculates faster than the 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 equipped with 12 GB of memory. In contrast, the Tesla K20 and GeForce GTX 980 each come with one GPU and less than 5 GB of memory. This directly affects how many model simulations can be accommodated. pPDA on the 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. Second, the maximum number of parallel threads in a computing block also affects speed. The earlier GeForce GT 720M GPU allows only a maximum of 512 parallel threads in a

block. We set a default launching block size at 32 parallel threads in order to accommodate these GPUs. Recent GPUs allow 1,024 maximum threads, so earlier GPUs, although they return 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, 1,024 parallel threads. This can easily be done by setting `nthread = 1024` in ppda's function calls. However, we have not yet thoroughly tested the influence of different block sizes on computation times, and CUDA programming involves many other intricacies related to the design of Nvidia GPUs and to parallel programming methods, such as “wrap divergence” (Cheng, Grossman, & McKercher, 2014, p. 82). In summary, a good and recent GPU, such as the 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

The pPDA technique, which is built on the foundation of previous KDE and PDA developments (Holmes, 2015; Parzen, 1962; Turner & Sederberg, 2014; Van Zandt, 2000), 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 a small investment (e.g., a PC equipped with a fast multicore CPU and Nvidia GPUs). The development of pPDA in the implementation adheres to the strict R package standard, making ppda and its source code accessible. In addition, we provide guidelines regarding how to apply PDA in Bayesian computation with ppda as a computational tool, allowing future researchers to explore a wide range of questions in cognitive process models.

Author note W.R.H. was supported by National Science Foundation (USA) Grant SES-1530760. A.H. is supported by Australian Research Council Discovery Project DP160101891

Appendix

The 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 1 accumulator (μ_{v_1}) and the Choice 2 accumulator (μ_{v_2}) in the first LBA process, the mean drift rates of the Choice 1 accumulator (μ_{w_1}) and the Choice 2 accumulator (μ_{w_2}) in the second LBA process, a common standard deviation of the drift rates for the two accumulators (σ), nondecision time (t_0), and a delay time (rD). Here we fixed σ at 1, as a constant, so the model had only the eight listed parameters.

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

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

At the first stage, the model draws the Choice 1 and Choice 2 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 means μ_{w_1} and μ_{w_2} .

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

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

v_1 and v_2 are the Choice 1 and Choice 2 drift rates for a trial before the switch time, and after the switch time these drift rates 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 would have sufficient trials to permit acceptable recovery of the PLBA parameters.

PLBA simulations

Method

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 s. For example, the upper left subplot in Fig. 13 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 curves in these panels hence show that 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 very large number of trials (16,384 per condition). We selected these parameter values because they generated data sets like the empirical data with responses that were influenced by both the preswitch and postswitch parameters. The first and the second mini-studies simulated $2^{14} = 16,384$, $2^{20} = 1,048,576$, $2^{24} = 16,777,216$, and $2^{27} = 134,217,728$ samples to synthesize PLBA probability density functions. We used these four cases to search for a minimal number of model simulation that would clearly locate the maximum likelihoods for all PLBA parameters. The aim of these two studies was to examine the influences of the trial

and model simulation numbers on estimating the PLBA likelihoods. Note that in the first mini-study we profiled the empirical data with arbitrarily chosen parameters, so the true value lines in the left panels of the figure [“Data (500–550 trials), assuming true values”] do not always match the maximum likelihood.

The third mini-study was a parameter recovery study. As is 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 was to test whether PLBA parameters would be 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 postswitch mean drift rates (Fig. 13). That is, if we used 1,048,576 model simulations, PDA might return imprecise μ_{w_1} and μ_{w_2} values corresponding to the maximum likelihoods. The very large numbers of trials and model simulations would minimize the influence of data and approximation noise, so if we failed to estimate PLBA parameters precisely, we could not hope to recover parameters in real data.

Results

The result of the first study indicated that when there are a similar number of data points in the empirical data, the likelihood maxima of the postswitch parameters are difficult to estimate, even with very large numbers of model simulations. The results of the second study showed that it is possible to locate clear maxima for those parameters with 16,384 data points and 2^{27} model simulations. Furthermore, the results suggested that with 16,384 trials and 2^{20} model simulations, there might still be some difficulty identifying rD . Although the results suggested that the ideal number of model simulations to identify maximum likelihoods is 2^{27} , this huge number of model simulations would 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 number of model simulations (see Fig. 12). The two studies together indicate that it is unlikely that postswitch mean drift rates would be identifiable with around 500 observations, regardless of how many model simulations were performed. Second, the approximation noise might hinder recovering postswitch parameters with tens of thousands of model simulations, which are often used in CPU-

Fig. 13 Likelihood profiles of the PLBA model. The profile plots examine whether the number of model simulations and the data points are sufficient to recover the parameters. We chose the parameters $A = 0.75, B = 0.25, \mu_{v1} = 2.5, \mu_{v2} = 1.5, \mu_{w1} = 1.2, \mu_{w2} = 2.4, rD = 0.1$, and $t_0 = 0.2$ in order to profile the likelihood. The log-likelihoods in the four left columns [Empirical data (500–550 trials), assuming true values] were calculated on the basis of the empirical data (only for the first participant). The four right columns [Simulation (16,384

trials)] are 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 with switch time fixed at 0.5 s (Holmes et al., 2016, calibrated this time for each participant). From left to right, the columns on the left and right sides of the figure show four different numbers of model simulations (16,384, 1,048,576, 16,777,216, and 134,217,728) for each PLBA parameter

based PDA. The results of the third study suggested that in the asymptotic case (huge numbers of trials and model simulations), all mean drift rates and A parameters could be

recovered with high precision, rD and t_0 estimates would be biased toward the upper end, and B estimates would be biased toward the lower end (Table 3).

Table 3 Point estimates and Bayesian credible intervals in the case of very large numbers of trials and model simulations

	A	B	μ_{v1}	μ_{v2}	μ_{w1}	μ_{w2}	rD	t_0
True value	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

References

- Ball, K., & Sekuler, R. (1982). A specific and enduring improvement in visual motion discrimination. *Science*, 218, 697–698. <https://doi.org/10.1126/science.7134968>
- Beaumont, M. A. (2010). Approximate Bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution, and Systematics*, 41, 379–406.
- Brooks, S., Gelman, A., Jones, G., & Meng, X. L. (2011). *Handbook of Markov chain Monte Carlo*. New York, NY: CRC Press.
- Brown, S. D., & Heathcote, A. (2008). The simplest complete model of choice response time: Linear ballistic accumulation. *Cognitive Psychology*, 57, 153–178. <https://doi.org/10.1016/j.cogpsych.2007.12.002>
- Cheng, J., Grossman, M., & McKercher, T. (2014). *Professional CUDA C programming*. Indianapolis, IN: Wiley.
- Chiu, S.-T. (1991). Bandwidth selection for kernel density estimation. *Annals of Statistics*, 19, 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, 11560–11571
- Dawson, M. R. (1988). Fitting the ex-Gaussian equation to reaction time distributions. *Behavior Research Methods, Instruments, & Computers*, 20, 54–57.
- Dutilh, G., Annis, J., Brown, S. D., Cassey, P., Evans, N. J., Grasman, R. P. P., . . . Donkin, C. (2018). The quality of response time data inference: A blinded, collaborative assessment of the validity of cognitive models. *Psychonomic Bulletin & Review*. Advance online publication. <https://doi.org/10.3758/s13423-017-1417-2>
- Gelman, A. (2014). *Bayesian data analysis*. Boca Raton, FL: CRC Press.
- Goldenshluger, A., & Lepski, O. (2011). Bandwidth selection in kernel density estimation: Oracle inequalities and adaptive minimax optimality. *Annals of Statistics*, 39, 1608–1632. <https://doi.org/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, 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, 678–694. <https://doi.org/10.3758/BF03206550>
- Heathcote, A., Brown, S. D., & Wagenmakers, E.-J. (2015). An introduction to good practices in cognitive modeling. In B. U. Forstmann & E.-J. Wagenmakers (Eds.), *An introduction to model-based cognitive neuroscience* (pp. 25–48). New York, NY, US: Springer Science + Business Media.
- Heathcote, A., Lin, Y.-S., Reynolds, A., Strickland, L., Gretton, M., & Matzke, D. (2018). Dynamic models of choice. *Behavior Research Methods*. Advance online publication. <https://doi.org/10.3758/s13428-018-1067-y>
- 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, 1593–1623.
- Hohle, R. H. (1965). Inferred components of reaction times as functions of foreperiod duration. *Journal of Experimental Psychology*, 69, 382–386. <https://doi.org/10.1037/h0021740>
- 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. <https://doi.org/10.1016/j.jmp.2015.08.006>
- Holmes, W. R., & Trueblood J. S. (2018). Bayesian analysis of the piecewise diffusion decision model. *Behavior Research Methods*, 50, 730–743. <https://doi.org/10.3758/s13428-017-0901-y>
- 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 (Working paper). Madison, WI: University of Wisconsin, Department of Statistics.
- Hu, B., & Tsui, K.-W. (2010). Distributed evolutionary Monte Carlo for Bayesian computing. *Computational Statistics and Data Analysis*, 54, 688–697. <https://doi.org/10.1016/j.csda.2008.10.025>
- Luce, R. D. (1986). *Response times*. New York, NY: 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, 798–817. <https://doi.org/10.3758/PBR.16.5.798>
- McClelland, J. L. (1979). On the time relations of mental processes: An examination of systems of processes in cascade. *Psychological Review*, 86, 287–330. <https://doi.org/10.1037/0033-295X.86.4.287>
- Miletić, S., Turner, B. M., Forstmann, B. U., & van Maanen, L. (2017). Parameter recovery for the Leaky Competing Accumulator model. *Journal of Mathematical Psychology*, 76, 25–50. <https://doi.org/10.1016/j.jmp.2016.12.001>
- Neal, R. M. (1994). An improved acceptance procedure for the hybrid Monte Carlo algorithm. *Journal of Computational Physics*, 111, 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>
- Palestro, J. J., Sederberg, P. B., Osth, A. F., van Zandt, T., & Turner, B. M. (2018). *Likelihood-free methods for cognitive science*. Cham, Switzerland: Springer.
- Parzen, E. (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33, 1065–1076.
- R Core Team. (2017). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from <https://www.R-project.org/>
- Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, 85, 59–108. <https://doi.org/10.1037/0033-295X.85.2.59>
- Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Computation*, 20, 873–922. <https://doi.org/10.1162/neco.2008.12-06-420>
- Robert, C. P. (1995). Simulation of truncated normal variables. *Statistics and Computing*, 5, 121–125. <https://doi.org/10.1007/BF00143942>
- Roberts, G. O., & Rosenthal, J. S. (2001). Optimal scaling for various Metropolis–Hastings algorithms. *Statistical Science*, 16, 351–367.
- Sanderson, C., & Curtin, R. (2016). Armadillo: A template-based C++ library for linear algebra. *Journal of Open Source Software*, 1(2), 26. <https://doi.org/10.21105/joss.00026>
- Schwarz, W. (2002). On the convolution of inverse Gaussian and exponential random variables. *Communications in Statistics Theory and Methods*, 31, 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*, 31, 93–99. <https://doi.org/10.2307/2347084>
- Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. London, UK: Chapman & Hall.
- Sisson, S. A., & Fan, Y. (2010). *Likelihood-free Markov chain Monte Carlo*. arXiv preprint. arXiv:1001.2058.
- Smith, P. L. (2016). Diffusion theory of decision making in continuous report. *Psychological Review*, 123, 425–451. <https://doi.org/10.1037/rev0000023>

- Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & van der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B*, 64, 583–639.
- Ter Braak, C. J. F. (2006). A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: Easy Bayesian computing for real parameter spaces. *Statistics and Computing*, 16, 239–249.
- Thura, D., Beauregard-Racine, J., Fradet, C.-W., & Cisek, P. (2012). Decision-making by urgency gating: Theory and experimental support. *Journal of Neurophysiology*, 108, 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, 28.
- Tsetsos, K., Usher, M., & McClelland, J. L. (2011). Testing multi-alternative decision models with non-stationary evidence. *Frontiers in Neuroscience*, 5, 63. <https://doi.org/10.3389/fnins.2011.00063>
- Turner, B. M., & Sederberg, P. B. (2012). Approximate Bayesian computation with differential evolution. *Journal of Mathematical Psychology*, 56, 375–385.
- Turner, B. M., & Sederberg, P. B. (2014). A generalized, likelihood-free method for posterior estimation. *Psychonomic Bulletin & Review*, 21, 227–250. <https://doi.org/10.3758/s13423-013-0530-0>
- 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, 368–384. <https://doi.org/10.1037/a0032222>
- Usher, M., & McClelland, J. L. (2001). The time course of perceptual choice: The leaky, competing accumulator model. *Psychological Review*, 108, 550–592. <https://doi.org/10.1037/0033-295X.111.3.757>
- Van Zandt, T. (2000). How to fit a response time distribution. *Psychonomic Bulletin & Review*, 7, 424–465. <https://doi.org/10.3758/BF03214357>
- Verdonck, S., Meers, K., & Tuerlinckx, F. (2016). Efficient simulation of diffusion-based choice RT models on CPU and GPU. *Behavior Research Methods*, 48, 13–27. <https://doi.org/10.3758/s13428-015-0569-0>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.