Title
Fast maximum likelihood estimation for general hierarchical models.

Journal
Journal of Applied Statistics, 52(3)

ISSN
0266-4763

Authors
Hong, Johnny
Stoudt, Sara
de Valpine, Perry

Publication Date
2025

DOI
10.1080/02664763.2024.2383284

Peer reviewed

JOURNAL OF APPLIED STATISTICS
2025, VOL. 52, NO. 3, 595–623
https://doi.org/10.1080/02664763.2024.2383284

Taylor & Francis
Taylor & Francis Group

Fast maximum likelihood estimation for general hierarchical models

Johnny Hong[a], Sara Stoudt [b] and Perry de Valpine[c]

[a]Department of Statistics, University of California Berkeley, Berkeley, CA, USA; [b]Department of Mathematics, Bucknell University, Lewisburg, PA, USA; [c]Department of Environmental Science, Policy and Management, University of California, Berkeley, California, USA

ABSTRACT

Hierarchical statistical models are important in applied sciences because they capture complex relationships in data, especially when variables are related by space, time, sampling unit, or other shared features. Existing methods for maximum likelihood estimation that rely on Monte Carlo integration over latent variables, such as Monte Carlo Expectation Maximization (MCEM), suffer from drawbacks in efficiency and/or generality. We harness a connection between sampling-stepping iterations for such methods and stochastic gradient descent methods for non-hierarchical models: many noisier steps can do better than few cleaner steps. We call the resulting methods Hierarchical Model Stochastic Gradient Descent (HMSGD) and show that combining efficient, adaptive step-size algorithms with HMSGD yields efficiency gains. We introduce a one-dimensional sampling-based greedy line search for step-size determination. We implement these methods and conduct numerical experiments for a Gamma-Poisson mixture model, a generalized linear mixed models (GLMMs) with single and crossed random effects, and a multi-species ecological occupancy model with over 3000 latent variables. Our experiments show that the accelerated HMSGD methods provide faster convergence than commonly used methods and are robust to reasonable choices of MCMC sample size.

ARTICLE HISTORY
Received 22 August 2021
Accepted 14 July 2024

KEYWORDS
Bayesian hierarchical models; maximum likelihood estimation; Markov chain Monte Carlo; Monte Carlo expectation maximization; Monte Carlo Newton-Raphson; stochastic gradient descent

1. Introduction

Hierarchical statistical models are widespread in the applied sciences because of their ability to capture complex relationships in data. In ecology, hierarchical models of species abundance through space and time have been applied to estimate species distributions and dynamics [62]. In political science, hierarchical models are used to estimate underlying preferences from data on policymaker decisions [35], while in epidemiology such models are used to estimate disease prevalence across space and time [46]. These models are difficult to estimate largely because the likelihood requires integration over the latent variables e.g. true occupancy across many sites for many individuals, political preferences of many states or policymakers, or the relative risk of disease across many areas, which is typically a high-dimensional problem with no closed-form solution [14,15].

CONTACT Johnny Hong jcyhong@berkeley.edu Department of Statistics, University of California Berkeley, 367 Evans Hall, Berkeley, CA 94720-3860, USA

Partly because of the difficulty of the likelihood integration problem, Bayesian analysis via computational tools such as Markov chain Monte Carlo (MCMC) has become the main practical path for analysis of many hierarchical models [16,33]. However, many lines of statistical reasoning would be enabled by a similarly general computational approach for maximum likelihood estimation [20]. One might seek to apply likelihood ratio tests, model selection by Akaike information criterion (AIC), goodness-of-fit as measured by maximum likelihood or other metrics, cross-validation, or other approaches. Although statisticians sometimes emphasize the philosophical incompatibility of Bayesian and frequentist results, practitioners are quite willing to study results from each side-by-side. Even when one seeks a Bayesian analysis, maximum likelihood results can provide a sanity check on the MCMC posterior and the influence of prior distribution assumptions. Statisticians have argued that the future will hold a combination of both Bayesian and frequentist methods [27], yet for general hierarchical models, practitioners in the applied sciences are often limited to Bayesian results. Amid this rich space for statistical innovation, the need for improved MLE methods for general hierarchical models is vital.

A variety of methods have been proposed for MLE estimation of general hierarchical models, but none has gained the kind of general traction that MCMC has for Bayesian estimation. One set of methods are stochastic variants on the expectation maximization (EM) algorithm [24], such as Monte Carlo EM (MCEM, [71]), stochastic EM (SEM, [12,13]), and stochastic approximation EM (SAEM, [11,23,43]). These suffer from the potentially slow convergence path of EM, and methods to ensure convergence involve costly increases in sample sizes to achieve smaller Monte Carlo variance as the algorithm proceeds [9]. Despite these issues, MCEM is one of the most widely applied methods because of its generality. A second approach, Monte Carlo Newton-Raphson (MCNR, [44]), also requires increasing Monte Carlo sample sizes to ensure convergence, although theory and application of this method appear less widespread. A third approach, data cloning [47] or State-Augmentation for Marginal Estimation (SAME, [36]), uses MCMC with many duplicate latent states, making the problem similar to MCMC but harder. A fourth approach, Monte Carlo Kernel Likelihood (MCKL, [19]), can require iterated application of MCMC.

Due to the various challenges in applying these methods, they are not used as widely as they might be. Specialized methods have been created for specific problems, such as stochastic approximation EM coupled with approximate Bayesian computation for state-space models (SAEM-ABC, [56]), stochastic approximation EM with a Metropolis-Hastings sampling procedure that is based on a multidimensional Gaussian proposal for nonlinear mixed effects models (f-SAEM, [38]), sequential Monte Carlo paired with stochastic approximation EM for working with functional data [48], and stochastic gradient MCMC for Gaussian Process computations [1], but these fail to cover a wide range of latent variable model scenarios and require specialized implementation of the algorithms. Finally, we note that there has been interest in distributional approximation methods such as Integrated Nested Laplace Approximations (INLA) [63] and variational Bayes [5,67] for the related Maximum Posterior Estimation problem, but we focus on problems where the goal is exact MLE, limited only by small Monte Carlo error.

In this paper, we make several contributions to the hierarchical model maximum likelihood problem. First, we exploit a connection to stochastic gradient descent methods (for non-hierarchical models) that sets up transferal of those methods to hierarchical maximum likelihood estimation. This expands the range of available methods for this important class

of problems. We refer to the methods in this new context as Hierarchical Model Stochastic Gradient Descent (HMSGD). Second, we place the new (for this context) methods and existing methods in a unified framework of iterative sample-move steps. This allows systematic comparison and potential combination of methods. Third, we show that Adaptive Moment Estimation (Adam) [41], a state-of-the-art step-size adaptation method from stochastic gradient descent problems, performs best in a set of case studies. Since stochastic gradient descent methods have been almost entirely unused and, to the best of our knowledge, Adam has never been used for hierarchical model maximum likelihood estimation, this establishes a highly competitive method that is new to this problem space. Fourth, we introduce a new method, called iterative 1D sampling, that also performs very well in comparisons. Fifth, we compare numerous methods, both well-known and newly exploited, to a series of empirical examples increasing in data size and model complexity. We find that Adam and the iterative 1D sampling methods – both newly adopted and/or developed here – perform best overall, giving efficiency improvements that range from 5-fold to 10-fold. Sixth, we give a simulation study comparing methods across a range of data sizes and the Monte Carlo sample sizes used iteratively within algorithms. This allows us to investigate how estimation precision and computational efficiency are related to these dimensions. Finally, we provide general implementations of all the methods in R package NIMBLE (Numerical Inference for statistical Models for Bayesian and Likelihood Estimation) [21,22].

Stochastic gradient descent problems and hierarchical maximum likelihood problems both involve large samples, but the samples are in different parts of the relevant models. In typical stochastic gradient descent problems, one has a non-hierarchical model for massive datasets ('big data', for example, more than a million observations) used in machine learning applications such as image recognition. For example, the loss function for neural network parameters is a sum of many terms, and the computation of its exact gradient to take an optimization step is costly [6,7]. It turns out to be more efficient to calculate a gradient from a stochastic subset of the data at each iteration and to use a running average of steps to smooth over the stochasticity [6,31,32]. In essence, many fast, noisy steps converge more efficiently than few slow, deterministic steps.

In a hierarchical model, the relevant gradient is an expectation over latent states, often approximated by Monte Carlo. In this context, the large sample is not the data but rather the Monte Carlo sample of latent states given parameters and data. Existing approaches represent different ways of taking approximately deterministic steps at the cost of large Monte Carlo sample sizes. The connection to stochastic gradient descent methods suggests that many faster but noisier steps may work better. We take advantage of the fact that highly developed step-size schedules from the stochastic gradient descent literature are directly transferable to the hierarchical model maximum likelihood problem in ways that have not been done before. The general view of the problem also leads us to propose the iterative 1D sampling method, a new method based on a greedy line search in an approximate gradient direction. This turns out to be better than previous methods but not best in our computational experiments.

While the connection between maximum likelihood estimation of hierarchical models and stochastic gradient descent of non-hierarchical models has been noted before in the computer science and machine learning literature [65,66] and discussed in the specific context of Hidden Markov Models [3,10], it has hardly been exploited at all. By placing

methods in a common framework, we can see them as variants on how to iterate between sampling latent states given data and current parameters and making a step to update parameters. The methods differ in sample-size and step-size choices, which we show can be improved by drawing on advances in stochastic gradient descent methods. (Despite that the likelihood is being maximized, we stick with the established label 'descent', viewing the negative log likelihood as a loss function to be minimized.) In addition to methods mentioned above, we include fixed step-size schedules in comparisons.

The new method we propose, called iterative 1D sampling, emerges naturally as a combination of the gradient-descent view of the problem and the Monte Carlo Kernel Likelihood idea. The essential idea is to draw MCMC samples of the latent states and parameters, but with parameters constrained to move in the direction of the approximated gradient at the values from the previous maximization. The new maximizer in this direction is then approximated using kernel density estimation (Figure 1). This can be viewed as a greedy line-search method. Section 3.3.3 describes this approach in detail.

We implement all of these methods in NIMBLE, [21,22]). NIMBLE is an R package that allows for flexible hierarchical model specification and writing algorithms such as MCMC or the methods proposed here. The system is extensible and automatically generates model- and algorithm-specific C++ for fast execution. We used a pre-release version of NIMBLE with support for automatic differentiation, from which all derivatives below are obtained,

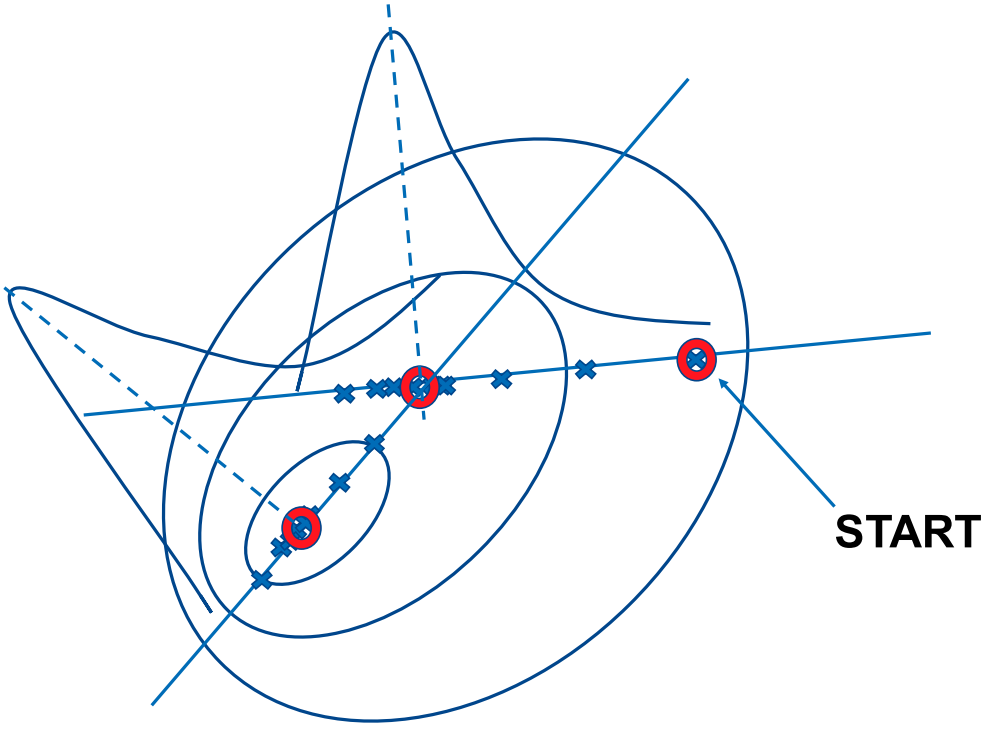

Figure 1. Visualization of the 1-dimensional sampling. The ellipses represent the contours of the likelihood surface. The blue crosses indicate the MCMC samples and the blue curves represent the density estimates. Each of the red circles indicates the parameter estimate at an iteration of the algorithms, which is computed as the mode of the estimated density.

which has subsequently been released as of NIMBLE version 1.00. This implementation can be straightforwardly used by practitioners requiring only a model specification to define their own latent variable problem of interest.

The rest of the paper is organized as follows. In Section 2, we establish notation for a general hierarchical model. In Section 3, we place MCEM, MCNR, and Hierarchical Model Stochastic Gradient Descent in a common framework and introduce Adam as a viable step-size schedule. We also introduce the iterative 1D sampling method. In Section 4, we discuss computational considerations, and in Section 5 we present computational experiments from four examples. The examples include a Gamma-Poisson mixture model of pump failure times, a logistic Generalized Linear Mixed Models (GLMM) with random intercepts for seed germination, a logistic GLMM for salamander mating success with crossed random effects (these present a challenge to many numerical methods), and a multi-species occurrence model with more than 3000 latent variables. Additionally, for the salamander example, we present simulation experiments for varying MCMC sample sizes and scientific sample sizes. Results show that Adam and the newly proposed iterative 1D sampling can achieve reasonably accurate estimates even with modest MCMC sample sizes, leading to improvement in computational time. Section 6 provides discussion and directions for future work.

2. Data-Generating process for hierarchical model

Suppose we have n observations $y = (y_1, \ldots, y_n) \in \mathcal{Y} \subset \mathbb{R}^n$, drawn from probability distribution $p(y \mid \theta)$, where $\theta = (\theta_1, \ldots, \theta_D) \in \Theta \subset \mathbb{R}^D$ are the model parameters. We introduce the latent variables $x = (x_1, \ldots x_K) \in \mathcal{X} \subset \mathbb{R}^K$, which are considered unobserved random variables. The general latent variable model structure is as follows: $x \mid \theta \sim p(x \mid \theta)$; $y \mid x, \theta \sim p(y \mid x, \theta)$.

The (marginal) likelihood of θ is $L(\theta) := p(y \mid \theta) = \int p(y \mid x, \theta) p(x \mid \theta)\, \mathrm{d}x$ and our goal is to find $\hat{\theta}^{ML} := \arg\max_{\theta \in \Theta} \log p(y \mid \theta)$. We are often interested in the scenario where the dimension of the latent variables is much larger than the dimension of the parameters; i.e. when $D \ll K$. When the dimension of the latent variables K is large, it is computationally infeasible to approximate the integral using a grid-based numerical integration. On the other hand, if D is small (say $D = 2$), it is tempting to use the naive Monte Carlo approximation $\frac{1}{S} \sum_{i=1}^{S} p(y \mid x^{(s)}, \theta)$ based on $x^{(s)} \sim p(x \mid \theta)$ for a grid of values of θ and find the maximizer. However, this rarely works well due to high variance, since the $x^{(s)}$ are drawn from a distribution without information from the data y. To remedy the high variability, the sample size S has to be large, which in turn increases the computational cost dramatically.

3. A general framework for sampling-based optimization approaches

We begin by giving the general framework within which MCEM, MCNR, and gradient descent are special cases.

Given a current iterate $\theta^{(t)}$, each of the algorithms performs the following two steps:

(1) *Sample Step:* Generate MCMC samples $x^{(t)} = (x^{(t),1}, x^{(t),2}, \ldots x^{(t),S})$ from $p(x \mid y, \theta^{(t)})$.
(2) *Move Step:* Update $\theta^{(t+1)} = f(x^{(t)})$.

where the choice of f is different for each algorithm. The notation we use emphasizes the dependence of f on the MCMC sample $x^{(t)}$, but f can also depend on any quantities involved in the computation up to iteration t.

For the sample step, it is important to note that MCMC sampling of latent states, with parameters held fixed, is typically much faster and better mixing that MCMC sampling of latent states and parameters jointly, i.e. when doing fully Bayesian estimation.

For the move step in most sampling-based approaches, gradient computation of the marginal log-likelihood is required. This can be seen from from Fisher's identity [26]:

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\log p(y\,|\,\theta) = \mathbb{E}_{X\sim p(x\,|\,y,\theta)}\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\log p(X,y\,|\,\theta)\right]. \tag{1}$$

Computation of the gradient of the complete log-likelihood (inside the expectation) can be done efficiently via an autodifferentiation package, and the expectation can be approximated via a Monte Carlo method. Specifically, the average of the gradient of the complete log-likelihood over a sample from $p(x\,|\,y,\theta)$ (obtained by MCMC or another method) approximates the gradient of the marginal log-likelihood.

Higher-order derivatives can be computed in a similar fashion [10]. For example, the Hessian of the log-likelihood admits the following representation

$$\begin{aligned}\frac{\mathrm{d}^2}{\mathrm{d}\theta\mathrm{d}\theta^T}\log p(y\,|\,\theta) &= \mathbb{E}_{X\sim p(x\,|\,y,\theta)}\left[\frac{\mathrm{d}^2}{\mathrm{d}\theta\mathrm{d}\theta^T}\log p(X,y\,|\,\theta)\right]\\ &+ \mathbb{E}_{X\sim p(x\,|\,y,\theta)}\left[\left(\frac{\mathrm{d}}{\mathrm{d}\theta}\log p(X,y\,|\,\theta)\right)\left(\frac{\mathrm{d}}{\mathrm{d}\theta}\log p(X,y\,|\,\theta)\right)^T\right]\\ &- \left(\frac{\mathrm{d}}{\mathrm{d}\theta}\log p(y\,|\,\theta)\right)\left(\frac{\mathrm{d}}{\mathrm{d}\theta}\log p(y\,|\,\theta)\right)^T,\end{aligned} \tag{2}$$

which is often known as Louis' identity [49]. This will prove useful in MCNR as discussed in Section 3.2.

When derivatives are not available, one can use a finite-element approximation of (1) [2]:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\theta}\log p(y\,|\,\theta) &= \frac{1}{p(y\,|\,\theta)}\frac{\mathrm{d}}{\mathrm{d}\theta}p(y\,|\,\theta)\\ &\approx \left(\frac{\frac{p(y\,|\,\theta+\delta e_1)}{p(y\,|\,\theta)}-1}{\delta},\ldots,\frac{\frac{p(y\,|\,\theta+\delta e_D)}{p(y\,|\,\theta)}-1}{\delta}\right),\end{aligned}$$

where e_i denotes the unit vector in the ith coordinate and δ denotes a very small value, say 10^{-4}. This suggests that the key to the approximation is to estimate the ratio $\frac{p(y\,|\,\theta+\delta e_i)}{p(y\,|\,\theta)}$. Since $\frac{1}{p(y\,|\,\theta)} = \frac{p(x\,|\,y,\theta)}{p(x\,|\,\theta)p(y|x,\theta)}$ for any x, we can express a likelihood ratio as

$$\frac{p(y\,|\,\psi)}{p(y\,|\,\theta)} = \frac{1}{p(y\,|\,\theta)}\int p(x\,|\,\psi)p(y\,|\,x,\psi)\,\mathrm{d}x = \int \frac{p(x\,|\,\psi)p(y\,|\,x,\psi)}{p(x\,|\,\theta)p(y\,|\,x,\theta)}p(x\,|\,y,\theta)\,\mathrm{d}x.$$

This can be approximated by an average of $\frac{p(x\,|\,\psi)p(y\,|\,x,\psi)}{p(x\,|\,\theta)p(y\,|\,x,\theta)}$ over a sample from $p(x\,|\,y,\theta)$. Letting $\psi = \theta + \delta e_i$ for $i = 1,\ldots,D$ provides approximation of the likelihood ratios needed for the finite-element approximation of the gradient.

Unfortunately, finite-element approximations of higher-order derivatives require division by extremely small values and hence are numerically unstable. In a preliminary version of this work, we used finite-element approximation for Hessians in MCNR and found that the algorithm often diverged very quickly in our numerical studies. In particular, parameters often jumped to the boundary of the search space in early iterations. Numerical instability of finite-element approximation of Hessians have been observed in the literature (for example, [55]). This issue is no longer present once derivatives are calculated accurately, which is done here via automatic differentiation.

Now we present MCEM, MCNR, and Stochastic Gradient descent and show how they fit into this unifying framework. Each method uses the same sample step, but the function f for the move step varies. We will refer to $G_{MC}(\theta, x)$ and $H_{MC}(\theta, x)$ as the Monte Carlo approximation of (1) and (2) at θ based on a sample x.

3.1. Optimization as the move step

The MCEM algorithm [71] replaces the expectation in the E-step of the traditional EM algorithm [24] with a Monte Carlo approximation. The corresponding move step is

$$f(x^{(t)}) = \arg\max_{\theta} \frac{1}{S} \sum_{i=1}^{S} \log p(x^{(t),i}, y \mid \theta). \tag{3}$$

3.2. Second order move step

Monte Carlo Newton-Raphson (MCNR) uses Monte Carlo estimates of the gradient and the Hessian of $\log p(y \mid \theta)$ for Newton-Raphson updates [44,51]. The corresponding move step is

$$f(x^{(t)}) = \theta^{(t)} - [H_{MC}(\theta^{(t)}, x^{(t)})]^{-1} G_{MC}(\theta^{(t)}, x^{(t)}). \tag{4}$$

3.3. First order move step

The first-order move step is obtained by replacing the Hessian in (4) with a step-size choice α. To allow each component to have its own step-size, we consider α to be a D-dimensional vector, where D is the number of components in the parameter vector. Denoting the Hadamard (component-wise) product as $\odot$, the first order move step can be written as

$$f(x^{(t)}) = \theta^{(t)} - \alpha \odot G_{MC}(\theta^{(t)}, x^{(t)}). \tag{5}$$

We will now describe three ways to select the step-size: fixed step-size, Adam, and one-dimensional greedy line search.

3.3.1. Fixed step-size

The fixed step-size method suggests the use of a fixed learning rate for each component. Tuning the size for a particular problem can be tricky to automate, so we appeal to other choices of α that rely less on a user explicitly tuning the method in the following subsections.

3.3.2. Adam

Adam [41] uses bias-corrected moment estimates of gradients. Instead of using the estimated gradient $G_{MC}(\theta^{(t)}, x^{(t)})$ at the current iterate, the move is governed by an adjusted gradient. For $i = 1, \ldots, D$, define the running averages of first and second-order moment estimates of the gradient: $m_i^{(t+1)} = \beta_1 m_i^{(t)} + (1-\beta_1)[G_{MC}(\theta^{(t)}, x^{(t)})]_i$ and $v_i^{(t+1)} = \beta_2 v_i^{(t)} + (1-\beta_2)([G_{MC}(\theta^{(t)}, x^{(t)})]_i)^2$, where β_1, β_2, α and ϵ are predetermined fixed scalars, and $m_i^{(0)}$ and $v_i^{(0)}$ are set to zero. The α term controls the step size; if chosen to be too big, we might not achieve convergence; if too small, it might take many steps to achieve convergence. Choosing $\beta_1, \beta_2 \in [0, 1)$ controls the exponential decay rates of moving averages of gradients and squared gradients in the algorithm. If both β_1, β_2 are chosen to be very close to 1 (i.e.: small decay rates), the first and second moment estimates can be stuck near 0. Following the recommendations in [41], we set $\beta_1 = 0.9$ and $\beta_2 = 0.999$ by default. While [41] recommends setting $\alpha = 0.001$ and $\epsilon = 10^{-8}$, we default $\alpha = 0.3$ and $\epsilon = 10^{-3}$ to ensure faster convergence and more stable estimate trajectory.

Define bias-corrected first and second-order moment estimates: $\hat{m}_i^{(t+1)} = \frac{m_i^{(t+1)}}{1-\beta_1^{t+1}}$; $\hat{v}_i^{(t+1)} = \frac{v_i^{(t+1)}}{1-\beta_2^{t+1}}$. The Adam update step is $f(x^{(t)}) = \theta^{(t)} - \alpha_{\text{adam}} \odot G_{MC,\text{adj}}^{(t)}$ with

$$\alpha_{\text{adam}} = \left[\frac{\alpha}{\sqrt{\hat{v}_1^{(t+1)}} + \epsilon}, \ldots, \frac{\alpha}{\sqrt{\hat{v}_D^{(t+1)}} + \epsilon} \right]$$

and $G_{MC,\text{adj}}^{(t)} = [\hat{m}_1^{(t)}, \ldots, \hat{m}_D^{(t)}]$.

In the context of stochastic gradient methods, convergence results are often stated in terms of bounds on the average regret, defined as $\frac{1}{T}[\sum_{t=1}^{T} f_t(\theta_t)] - \min_{\theta'} \frac{1}{T}[\sum_{t=1}^{T} f_t(\theta')]$ with f_t being a sequence of convex loss functions. [41] shows that Adam achieves an average regret bound of $O(1/\sqrt{T})$ under mild conditions, one of which is the convexity of the objective function. While in general a hierarchical model might not be globally convex, provided that the sample size (of y) is large enough, often the likelihood surface is locally similar to a Gaussian shape near the optimum and hence locally convex.

3.3.3. One dimensional greedy line search

The idea of using an adaptive step-size has been explored in stochastic approximation and gradient methods [57,75]. We introduce a novel adaptive step-size method called *1D greedy line search* that has its roots in Monte Carlo likelihood estimation by weighted posterior kernel densities (MCKL, [19]). The idea is to step to the maximum in the direction of the gradient by solving the following optimization problem,

$$c_t^{\max} = \arg\max_c p(y \mid \theta^{(t)}) + c G_{MC}(\theta^{(t)}; x^{(t)}), \tag{6}$$

where c is a scalar multiplying the current gradient with respect to θ. In essence G_{MC} is a single new parameter axis, along the current gradient, and c is the coordinate on that axis. Hence $c_t^{\max}$ is the one dimension MLE in that axis. Then there is an update step:

$\theta^{(t+1)} = f(x^{(t)})$, where

$$\alpha_{1D}^{(t)} = [c_t^{\max}, \ldots, c_t^{\max}]$$

and $f(x^{(t)}) = \theta^{(t)} + \alpha_{1D}^{(t)} \odot G_{MC}(\theta^{(t)}; x^{(t)})$.

In essence we are always choosing the 'best' step-size in the sense that we pick the one that provides the most progress. To approximately solve the optimization problem, given $(x^{(t)}, \theta^{(t)})$, we sample jointly in (x, γ) from $\tilde{p}(x, \gamma \mid y) \propto p(x, \theta^{(t)} + \gamma\, G_{MC}(\theta^{(t)}; x^{(t)}) \mid y)$. We are again reducing the parameter space to a single axis along the current gradient, here with coordinate γ, and now obtaining a posterior sample for (x, γ). The unrestricted samples of γ approximate a one-dimensional slice of the marginal distribution. We approximate the maximizer on the line using a kernel density estimate of the MCMC samples.

The advantage of using a 1D line search is the potential of aggressive moves at the start of the algorithm. The downside of 1D line search is the computational cost incurred by additional MCMC sampling at each step. In addition, the number of MCMC samples needed for the greedy line search has to be reasonably large in order for the kernel density estimation (and hence the mode estimation) to be reliable.

Algorithm 1 Gradient descent via 1D Greedy Line Search

- **Input:** $\theta^{(t)}, g^{(t)} := G_{MC}(\theta^{(t)}; x^{(t)})$.

1: Run an MCMC sampler to sample $(x^{(t)}, \gamma^{(t)})$ from $\tilde{p}(x, \gamma \mid y) \propto p(x, \theta^{(t)} + \gamma\, g^{(t)} \mid y)$.
2: Perform a 1D kernel density estimate for the sampled $\gamma^{(t)}$ and compute the mode $\hat{\gamma}^{(t)}$.
3: Set $\theta^{(t+1)} = \theta^{(t)} + \hat{\gamma}^{(t)} g^{(t)}$.

3.3.4. Other potential approaches

We note that two common step-size choices are not useful for our problem. The first is an inexact line search based on Wolfe conditions. Wolfe conditions guarantee sufficient improvement in the iterate and a decrease in the magnitude of the gradient [72]. Unfortunately, checking Wolfe conditions in our context is computationally costly, since it requires multiple evaluations of the marginal likelihood at each iteration. The second is the Robbins-Monro step-size schedule [60], a popular step-size schedule for root finding, applications in regression [40], probability density estimation [39], and stochastic gradient methods in training neural networks. However, we found in our preliminary experiments that the Robbins-Monro step-size in our context leads to slow convergence compared to the alternative step-size schedules we considered and requires careful tuning. We therefore omit the Robbins-Monro results below.

4. Computational considerations

4.1. Burn-In and warm start

For each iteration, we set the burn-in of the sample step to be half of the MCMC samples to be conservative (Section 6.5 of [8]). Since diagnostics require considerable amount of time to run and assess, it is more beneficial to run more samples (and conservatively remove

the first half of the samples) rather than worry about tracking diagnostics throughout each step. We also implement a 'warm start' where the last draws from the previous iteration's sample step are the starting point for the next iteration. This idea is used in many contexts [18,61,70] and in our case should also help ensure that the burn-in period is adequate. We leverage the progress from the previous iteration, taking advantage of the proximity of the parameter estimates between two adjacent iterations.

4.2. Kernel and bandwidth for the 1D sampling approach

For the kernel density estimation involved in the 1D sampling, we find that almost any standard choice of kernel (e.g. Gaussian, Epanechnikov, etc.) and the bandwidth (e.g. Silverman's rule-of-thumb [64]) yields similar final maximum likelihood estimates. It has been noted that the differences in statistical efficiency among these kernels are small [69]. For the numerical experiments, the kernel choice is defaulted to be Gaussian and the (optimal) bandwidth is computed based on the effective MCMC sample size instead of the nominal sample size, accounting for the fact that the usual optimal bandwidth is derived based on an independently identically distributed (i.i.d.) assumption [64].

4.3. MCMC sample sizes

Just as the number of MCMC samples that we use for the gradient estimation is a tuning parameter, analogous to the batch size in stochastic gradient descent, so is the number of MCMC samples for 1D sampling. For the 1D sampling, a larger MCMC sample size could potentially be necessary due to the slow mixing of joint sampling of the parameters and latent variables. In our experiments, the MCMC sample size choice[1] of 300 seems to work reasonably well for both gradient estimation and 1D sampling. In one of our case studies, reasonably accurate performance can still be achieved with only a sample size of 20.

4.4. Convergence criteria

For stochastic gradient descent and its variants, the convergence is often checked by predictive performance in machine learning applications. This is not appropriate in our MLE context since we are not solving a prediction problem. Within MCNR, [44] uses a formal hypothesis testing procedure where the variance of the updates are deduced based on a bootstrapping procedure. This is not applicable to our framework since the sample sizes involved in our algorithms tend to be too small for proper variance estimation. Lastly, a less formal option for convergence check is to plot the trajectory of the iterates and see if the trajectory for each parameter roughly fluctuates around a particular number [9]. This requires users to study the trajectory plots, which is not ideal for an automated MLE algorithm.

Our approach is to quantify the fluctuation and flatness of the estimate trajectory, leading to the development of a two-step test for convergence. This approach is chosen to be ad hoc but fast. Once the two-step test is passed, the algorithm is terminated. For the first step, we use the Wald-Wolfowitz runs test [68] to determine whether the trajectory is close to being monotonic, an indication of non-convergence. More specifically, at the tth

iteration of the algorithm, check if the number of runs, groups of consecutively increasing or decreasing coordinate values, in $(\theta_j^{(t-w+1)}, \theta_j^{(t-w+2)}, \ldots, \theta_j^{(t)})$ are at least $r = 4$ for a block size $w = 20$. (The choices of r and w are somewhat arbitrary here and can be tuned.) If this is not the case for every coordinate, continue the algorithm. If all the coordinates have runs of at least r, we proceed to the next test. For the second step, we carry out a t-test to compare the average of iterates in the most recent 20 iterations with that in the preceding 20 iterations. If we fail to reject the null hypothesis at a certain significance level (say 30%), we terminate the algorithm and conclude convergence. Note that a larger significance level means we are being conservative, and therefore we will favor running more iterations. In some examples, our ad hoc test is conservative, not determining that convergence has occurred and hence making methods look slower than they might be.

We remark that for MCEM in our numerical experiments, we use the implementation in NIMBLE, which is based on work by Caffo [9]. The implementation gradually increases the MCMC sample sizes, unlike our proposed approach of fixing a small MCMC sample size. Since the gradient estimates are reliable for large MCMC sample size, this allows the simple convergence check: check whether the approximated gradient is within a certain tolerance. This MCEM approach increases computation time, and in general convergence criteria are an important area for further research.

5. Numerical experiments

We experiment with the algorithms using four examples: a conjugate Gamma-Poisson hierarchical model (referred to as *pump*), two GLMMs (referred to as *seeds* and *salamander*) and a multi-species occurrence model with a large number of latent variables. These examples were chosen to assess the performance of our approach across a range of potential pitfalls that practitioners may encounter. The first three examples can be solved by specialized methods to give a correct MLE for comparison from the much more general methods studied here. For gradient and Hessian computations, we use automatic differentiation instead of finite-element approximation for better speed and higher accuracy. We run each algorithm for 300 iterations and report the execution times. In addition, if the algorithm passes the convergence test within 300 iterations, we report the convergence time and the number of iterations to convergence. To obtain the final estimates, we take the 20% trimmed mean of the last 20 iterates. The averaging is to smooth out any 'bouncing around' the optimum towards the end of the path, and the trimming is to make the estimate more robust to occasional deviations on the path. Similar stabilizing approaches exist in the stochastic gradient descent literature such as taking the average of the last α proportion of iterates, called α-suffix averaging [59].

For every algorithm in each example, we report the CPU execution time (for 300 iterations), the CPU time to convergence (that is, how long it takes until the convergence test passes), and the log-likelihood difference, compared to the benchmark estimates. We also compute the root-mean-squared error (RMSE) of the estimates of each of the HMSGD methods based on 30 random initializations to evaluate the robustness of each method. Detailed numerical results can be found in the Appendix. In some examples, we also qualitatively assess robustness of methods to different starting values based on estimate trajectories.

5.1. Case studies of different models

5.1.1. Case study: pump (Gamma-Poisson hierarchical model)

The *pump* model [30] is a classic example from WinBUGS [50]. It is a conjugate Gamma-Poisson hierarchical model, so the marginal likelihood can be analytically derived. The MLE can be found by a standard deterministic optimization procedure[2]. The model specification is as follows: for $i = 1, \ldots, N$, $\theta_i \mid \alpha, \beta \sim \text{Gamma}(\alpha, \beta)$; $\lambda_i = \theta_i t_i$; $x_i \mid \lambda_i \sim \text{Poisson}(\lambda_i)$, where x_i is the number of failures for pump i, θ_i is the failure rate for pump i, and t_i is the length of operation time for pump i. We treat $x_1, \ldots, x_N$ as the observed random variables, $t_1, \ldots, t_N$ as fixed constants, and $\theta_1, \ldots, \theta_N$ as latent variables. The pump reliability dataset is originally from [28]. It consists of data about ten power plant pumps ($N = 10$).

To investigate the sensitivity of the algorithms to initial values, each of the algorithms is tested with two different starting points $(\alpha^{(0)}, \beta^{(0)}) = (10, 10)$, relatively far from the MLE, and $(\alpha^{(0)}, \beta^{(0)}) = (10, 2)$, unbalanced in its distance from the MLE. We observe that all the algorithms except the smaller fixed step-size (0.005) are able to get close to the benchmark MLE. Figure 2 shows that the 1D sampling approach makes aggressive moves initially, so it gets close to the optimum in fewer iterations. The smaller fixed step-size (0.005) follows the shape of the likelihood surface more closely at the cost of many more steps. Adam's final iterations concentrate more tightly around the optimum. The methods are in general robust to both different initial values and different MCMC sample sizes. The latter robustness allows us to reduce the computational time by not relying on as many MCMC samples to assure good performance. From the RMSE perspective (Tables A5 and A6), Adam, Newton-Raphson, and 1D sampling provide more accurate estimates compared to the two fixed step-size schedules. MCEM takes far fewer iterations but a much longer computational time to reach the optimum.

5.1.2. Case study: seeds (logistic regression with random effects)

Our next example, *seeds*, is a logistic regression model with random effects. These types of models are common in social sciences and medicine as many longitudinal studies have a binary outcome [54]. The *seeds* example has appeared in [50] as a classic WinBUGS example, and the dataset is originally from [17].

The model specification is as follows: $\beta_{2i} \sim \text{N}(0, \sigma^2_{RE})$; $\text{logit}(p_i) = \beta_0 + \beta_1 x_i + \beta_{2i}$; $r_i \sim \text{Binom}(n_i, p_i)$, where r_i is the binary outcome of interest for individual i, the x_i are the explanatory data collected per individual i, the n_i are the number of replications per individual i, and the β_{2i} are the unobserved random effects for individual i. We treat $x_1, \ldots, x_N$ as the observed random variables, $n_1, \ldots, n_N$ as fixed constants, and the β_{2i} and p_i as latent variables. The parameters of interest are β_0, β_1, and σ_{RE}. Our dataset consists of twenty one individuals ($N = 21$).

We experiment with two sets of initial values, $(\beta_0, \beta_1, \sigma_{RE}) = (0, 0, 1)$ and, further from the MLE, $(\beta_0, \beta_1, \sigma_{RE}) = (-1, -1, 4)$, as well as two different MCMC sample sizes, 20 and 300. With MCMC sample size 300 and initial value $(-1, -1, 4)$, Newton-Raphson seems to be stuck at the boundary constraints and fails to get close to the benchmark MLE, showing that Newton-Raphson could be sensitive to initial values.

Figure 3 displays the trajectories of the iterates for each algorithm with MCMC sample size 300 and initial value $(0, 0, 1)$. The larger fixed step-size is sensitive to large

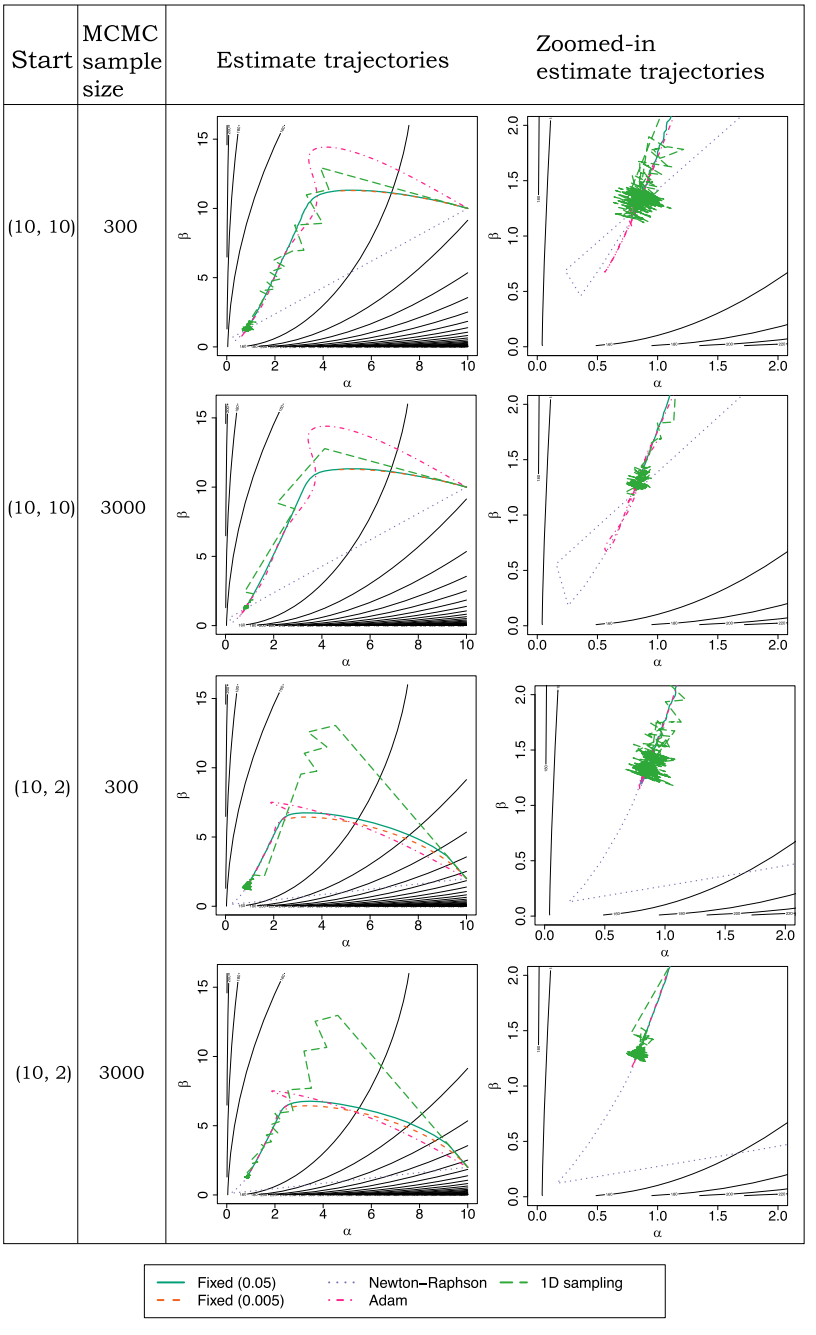

Figure 2. Estimate trajectories for the *pump* example. Zoomed-in trajectories are based on the final 100 iterations of the algorithms. The true MLE is $(\hat{\alpha}, \hat{\beta}) = (0.823, 1.262)$.

gradients, leading to erratic jumps. Contrasting with the *pump* example, here the smaller step-size is preferable, suggesting that the fixed step-size method might require a careful step-size choice to achieve well-behaved trajectories. All of the other methods appear to converge within a narrow band around the true parameters fairly quickly. Adam and

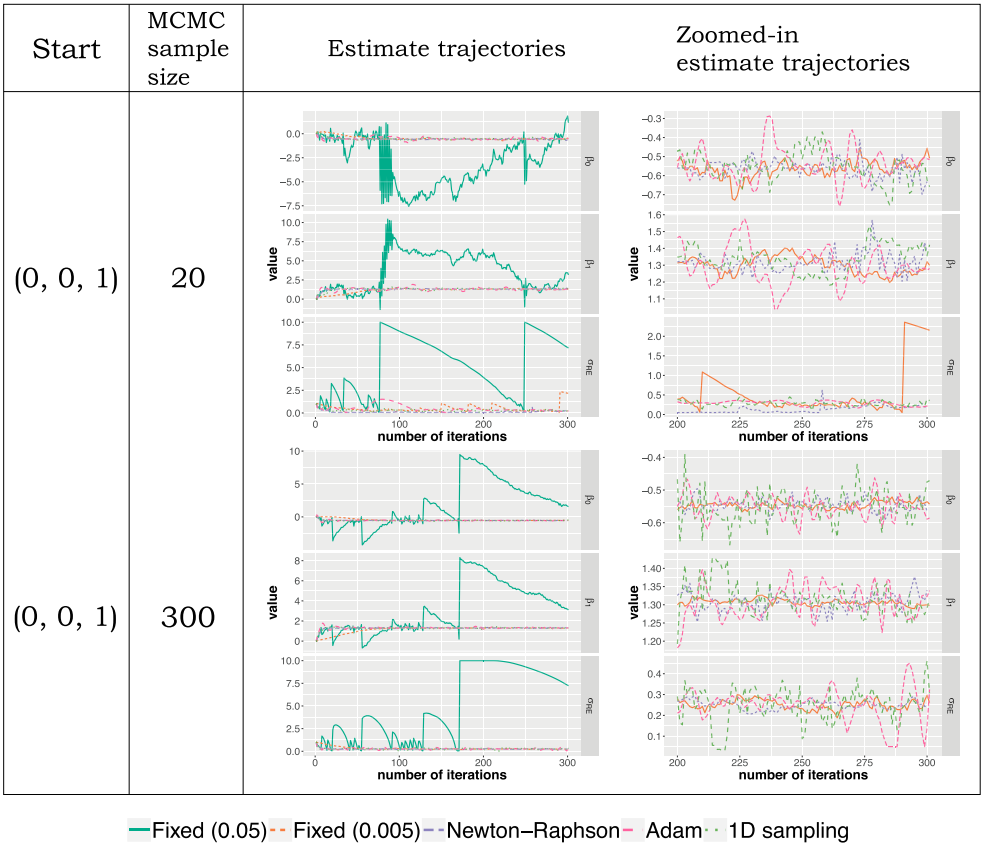

Figure 3. Estimate trajectories for the *seeds* example. Zoomed-in trajectories are based on the final 100 iterations of the algorithms. The fixed step-size (0.05) trajectories are dropped in the zoomed-in plots for better resolution of the other methods. The *lme4* estimates suggest that a decent ML estimate should be around $(\hat{\beta}_0, \hat{\beta}_1, \hat{\sigma}_{RE}) = (-0.548, 1.310, 0.249)$.

Newton-Raphson still perform well with MCMC sample sizes of 20 (Table A7), but both the small fixed step-size and the large fixed step-size approaches have trouble with a small MCMC sample size (Table 3).

Comparing Table A9 with Table A7, we observe that for smaller MCMC sample sizes more iterations are typically needed to reach convergence. Adam arrived at a solution several times faster than 1D sampling or MCEM (Tables A7 and A9). The smaller fixed step-size approach also does fairly well. From the RMSE perspective (Tables A11 and A12), with MCMC sample size 300, both Adam and 1D sampling provide more accurate estimates compared to the fixed step-size schedules. On the other hand, Newton-Raphson diverges for certain random initializations, showing lack of robustness to starting values. In this case we can also compare to the results given by the specialized method in the lme4 package [4] in R, and we see close agreement in the estimates.

5.1.3. Case study: salamander (crossed random effects model)

Our next example, *salamander* [37], features a GLMM with crossed random effects, which lead to challenging estimation of the random effects' variances. Let y_i be the observed

outcome of whether salamander pair i successfully mated or not. For pair i, we use $F(i)$ and $M(i)$ to denote the corresponding female and male. Let $\text{REF}_{F(i)}$ and $\text{REM}_{M(i)}$ denote the random effects for female $F(i)$ and male $M(i)$. The model specification is as follows: $\text{REF}_{F(i)} \sim N(0, \sigma_F^2)$; $\text{REM}_{M(i)} \sim N(0, \sigma_M^2)$; $\text{logit}(\theta_i) = \beta_1\ \text{isRR}_i + \beta_2 \text{isRW}_i + \beta_3 \text{isWR}_i + \beta_4\ \text{isWW}_i + \text{REF}_{F(i)} + \text{REM}_{M(i)}$; $y_i \sim \text{Bern}(\theta_i)$, where isRR, isRW, isWR, and isWW encode which population the female (first letter) and male (second letter) are from in each pair with R denoting 'rough-butt' and W denoting 'whiteside'; θ_i is the probability of mating for each pair i. We consider the random effects $\text{REF}_{F(i)}$, $\text{REM}_{M(i)}$ as latent variables. The top level parameters of interest are $\beta_1, \beta_2, \beta_3, \beta_4, \sigma_F^2$, and σ_M^2. Data from 360 pairs of salamanders ($N = 360$) is available in the *glmm* package [42] in R.

We experiment with two initial values of the six parameters $(\beta_1, \beta_2, \beta_3, \beta_4, \sigma_F^2, \sigma_M^2)$, (2, 2, 2, 2, 2, 2) and, further from the MLE, (4, 4, 4, 4, 4, 4). Our benchmark estimate is from the *lme4* package [4] in R. In addition, we also compare our estimates with the ones from the *glmm* package [42]. We take the advice in the documentation of *glmm* to increase the Monte Carlo sample size to 10^5 from the default of 10^4 in order to get more reliable estimates of the parameters. We also study the robustness of starting values by randomly initializing 30 starting values. From the RMSE perspective (Table A15), fixed step-size (0.05), Adam, and 1D sampling perform favorably compared to fixed step-size (0.005); Newton-Raphson diverges for certain initializations.

From Figure 4, we observe that all the methods arrive at a stable estimate of the β values quickly when the initial value is (2, 2, 2, 2, 2, 2). In particular, Newton-Raphson and 1D sampling get close to the benchmark MLE within ten iterations. We compute the approximate log-likelihood values at the various MLEs via *lme4*. As shown in Tables A13 and A14 in the Appendix , the sampling based approaches yield MLEs close to the ones given by *glmm* and *lme4* in terms of both log-likelihood differences and RMSE, regardless of the initial values. However, the convergence criteria did not perform well and remain an area for future research.

5.1.4. Case study: multi-species occurrence model

Our final example is a multi-species, single-season occurrence model with more than 3000 latent variables and 20 parameters. This type of high-dimensional problem often has slow MCMC mixing. The full model specification can be found in (A3)-(A5) in Appendix 1 of Ponisio et al. [58], while the original analysis and data are from Zipkin et al. [76]. To ensure the parameter trajectories do not go out of range for the HMSGD methods, we apply an inverse-logit transformation for parameters with range $[0, 1]$ and an exponential transformation for parameters with range $(0, \infty)$ so that the range for each transformed parameter is $(-\infty, \infty)$.

For the HMSGD methods, we initialize all the transformed parameters to 0. From Tables A16, A17 and A18, fixed step-size (both 0.05 and 0.005) and Adam yield MLEs close to the one given by MCEM but with much shorter computational time. On the other hand, 1D sampling runs into trouble with the MCMC sampling after a few iterations and Newton-Raphson gives poor estimates for some variance components.

However, when the transformed parameters are randomly initialized, the enormous RMSEs of fixed step-size approaches indicate that the methods diverged far from the MLE (Tables A19 and A20). For example, the RMSEs of σ_{uCATO} for fixed step-sizes 0.05 and 0.05 are 4.5×10^9 and 454.217 respectively, illustrating that an ill-chosen step-size can lead to

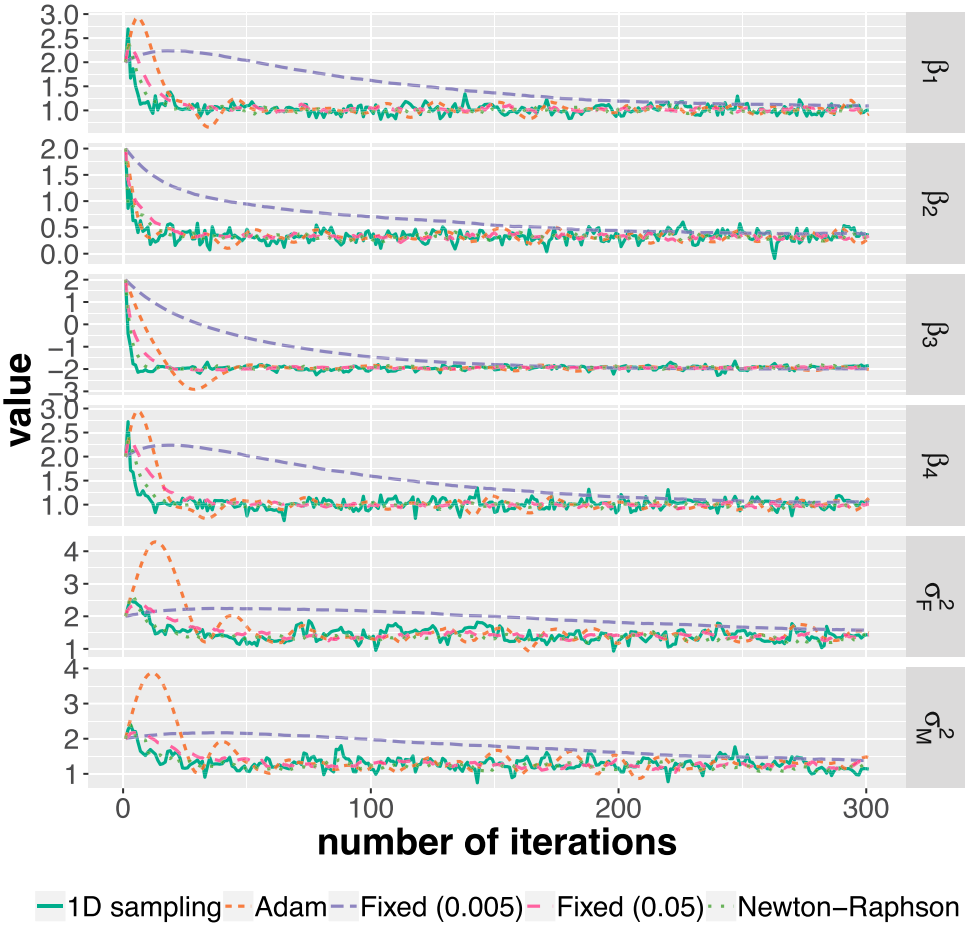

Figure 4. Estimate trajectories for the *salamander* example with MCMC sample size 300. Newton-Raphson and 1D sampling get close to the benchmark MLE within ten iterations, despite the fact that they do not pass the ad-hoc convergence criterion. The *glmm* estimates suggest that a decent ML estimate should be around $(\beta_1, \beta_2, \beta_3, \beta_4, \sigma_F^2, \sigma_M^2) = (1.023, 0.335, -1.908, 1.006, 1.326, 1.221)$, while the *lme4* estimates suggest that a decent ML estimate should be around $(\beta_1, \beta_2, \beta_3, \beta_4, \sigma_F^2, \sigma_M^2) = (1.008, 0.306, -1.896, 0.990, 1.174, 1.041)$.

very poor estimates; as a comparison, for the adaptive step-size approach Adam, the RMSE of σ_{uCATO} is only 19.31. Upon inspection of some estimate trajectories, we found that the variance estimates were often trapped at large values, most likely due to the underlying likelihood being close to flat in those regions. Results for this example again show a need for more refined stochastic convergence criteria.

5.2. Cases studies of varying MCMC sample sizes and scientific sample sizes

To understand the effect of varying MCMC sample sizes N_{mcmc} and scientific sample sizes N_{obs}, we conduct further simulation studies for the salamander example (Section 5.1.3; crossed random effects model). Recall that in the example $N_{mcmc} = 300$ and $N_{obs} = 360$.

We now vary $N_{mcmc} \in \{50, 100, 200, 300\}$ and $N_{obs} \in \{180, 360, 720, 1440\}$. To reduce N_{obs} from 360 to 180, we keep only the observations for the first 30 female salamanders. To double or quadruple N_{obs} from 360 to 720 or 1440, we clone the existing data and assign new indexes for the cloned observations. This holds the MLE unchanged while giving a sharper likelihood surface.

As the MCMC sample size N_{mcmc} decreases, the estimate trajectories have larger fluctuations, and hence convergence becomes less apparent (Figure 6). As expected, there is a linear relationship between the MCMC sample size and the execution time (Figure 5). However, the rate of change in execution time over MCMC sample size is higher for Newton-Raphson and 1D sampling than for fixed step size and Adam, due to the more involved computations needed for Newton-Raphson (Hessian matrix) and 1D sampling (kernel density estimation).

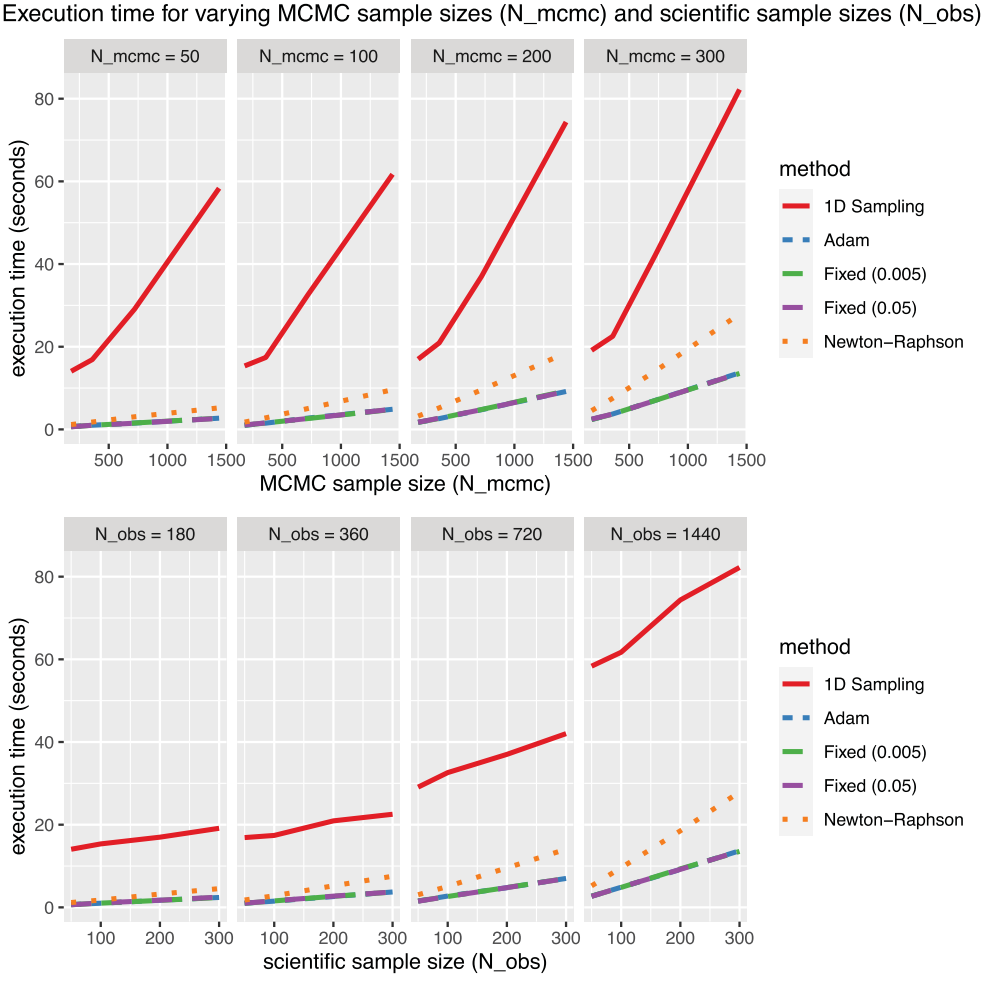

Figure 5. Execution times for varying MCMC sample sizes $N_{mcmc} \in \{50, 100, 200, 300\}$ and scientific sample sizes $N_{obs} \in \{180, 360, 720, 1440\}$ in the *salamander* example.

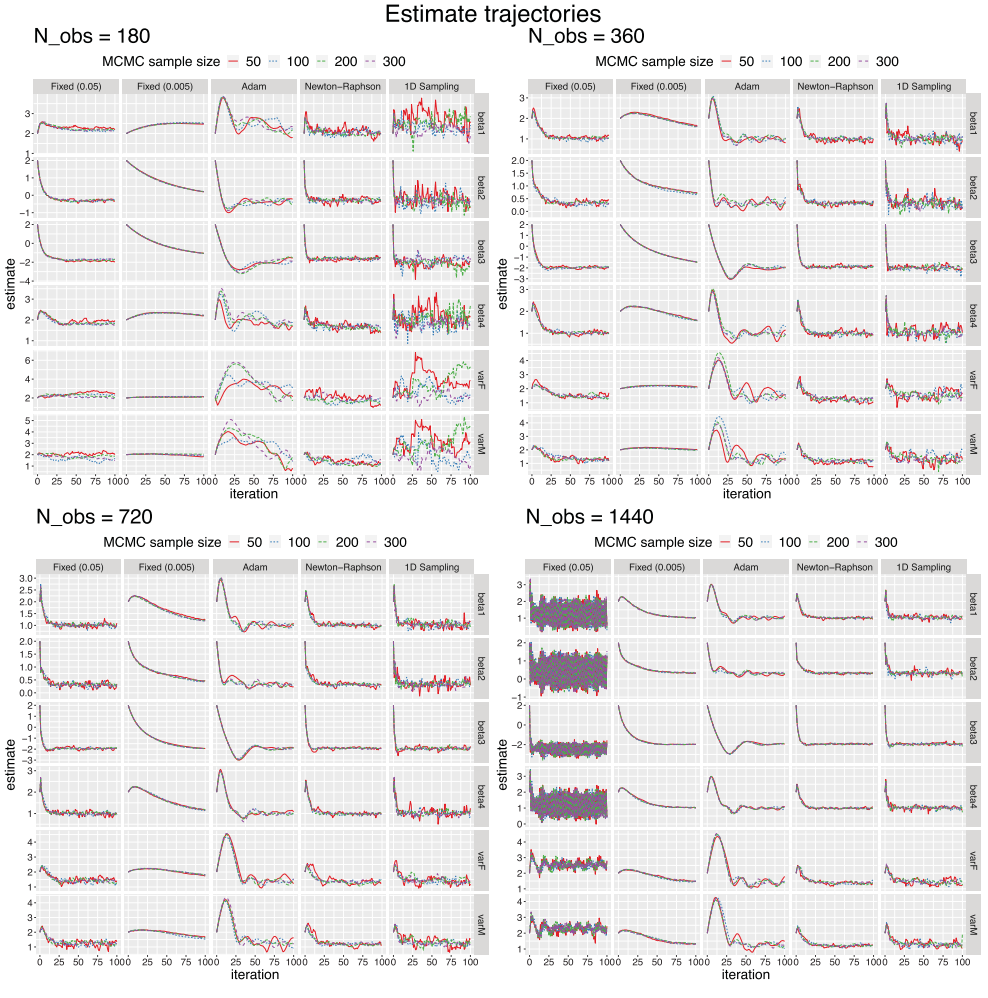

Figure 6. Estimate trajectories for varying MCMC sample sizes $N_{mcmc} \in \{50, 100, 200, 300\}$ and scientific sample sizes $N_{obs} \in \{180, 360, 720, 1440\}$ in the *salamander* example.

As the scientific sample size N_{obs} increases, the estimate trajectories look very similar (Figure 6). From the execution time perspective (Figure 5), the observations for varying N_{obs} are similar to to those from varying N_{mcmc}: a linear relationship between N_{obs} and the execution time, and a larger rate of change in execution time for Newton-Raphson and 1D sampling.

Notably for $N_{obs} = 180$ and small N_{mcmc} (top left of Figure 6), the 1D sampling estimate trajectories are highly volatile. The volatility might be due to kernel density estimation typically requiring a decently large sample size to be precise. We recommend that running 1D sampling with a reasonably large but modest MCMC sample size (that is, $N_{obs} = 300$) to ensure trajectory stability, or running 1D sampling for fast initial moves and then switching to other approaches. On the other hand, for $N_{obs} = 1440$ (bottom right of Figure 6), the large fixed step size estimate trajectories have large fluctuations around the correct MLE, suggesting that performance of fixed step sizes can be sensitive to sharpness of the likelihood surface.

6. Discussion and future work

We presented a unifying framework for various sampling-based MLE algorithms and proposed various extensions for fitting general hierarchical models with a large number of latent variables. In particular, we have experimented with the use of adaptive step-size schedule Adam in the context of MLE for hierarchical models, and introduced the 1D sampling approach as a viable step-size determination procedure. Our numerical experiments have shown promising results for various algorithms, especially Adam and 1D sampling, in terms of achieving short computational time and obtaining reasonable parameter estimates.

In preliminary work, we experimented with other adaptive step-size schedules, Adagrad [25] and Adadelta [74]. We found that Adagrad decays the step-size too aggressively and results in slow convergence; Adadelta often ends in highly oscillating behaviors near convergence. Hence we do not include the results from Adagrad and Adadelta in this work. We found that automatic differentiation not only speeds up gradient computations but also helps stabilize the Newton-Raphson method. Empirically, the stochastic gradient approach is robust to small MCMC sample sizes. We conjecture that this robustness is due to the fast mixing behavior of the MCMC when we are sampling the latent variables given the data and the top-level parameters.

With the use of a 'warm start' the algorithms might afford to have a much smaller burn-in, rather than half of the MCMC samples, since the starting point is reasonably close to the high density region of the stationary distribution. It would be interesting to formalize the benefits of a 'warm start' in the MCMC sampling part of the algorithms. An alternative approach suggested by a reviewer is to run several MCMC chains in parallel with random initializations and evaluate convergence based on the $\hat{R}$ diagnostic suggested by Gelman [29]. This approach is likely to yield better MCMC convergence at the cost of additional computations.

It is difficult, to establish theoretical comparisons among different sampling-based MLE approaches in terms of computational time. Due to difficulties in theoretical comparisons, we conducted experiments to investigate the performance of algorithms. Our experiments have two main limitations. The first limitation is that the conclusions might not be generalizable to all latent variable models. Except for the multi-species occurrence model, we have chosen examples where a benchmark MLE can be computed via specialized methods. For the *pump* example, we can explicitly find the marginal likelihood and compute the MLE; for the two GLMM example we use the results from *lme4* as benchmarks, although *lme4* relies on Laplace approximation and hence the results might not be accurate. The second limitation is that it is impossible to conduct experiments with all possible tuning parameter configurations for each of the MLE algorithms. Further work on tuning parameters for each of the methods could be useful.

However, we observe in numerical experiments that HMSGD with Adam and 1D sampling typically work well with default tuning parameters. In addition, their estimate trajectories are relatively stable compared to HMSGD with fixed step-size. A hybrid approach, such as running 1D sampling initially and then switching to MCNR or Adam, might be beneficial, although determining when to switch to the other algorithm can be tricky to automate. Similarly, we can view Algorithm 1 as a 'warm start' to make aggressive moves towards the MLE, and then switch to other algorithms with concrete theoretical guarantees.

Table 1. Current challenges for existing MLE methods and their solutions via HMSGD-based approaches.

Method	Challenge(s)	HMSGD-based solution
EM	Analytical derivation of E-step and/or M-step is required.	HMSGD-based solutions do not require analytical derivation.
MCEM	Fully-automated MCEM is computationally slow due to increasingly large MCMC sizes.	HMSGD with Adam can work reasonably well with small MCMC sample sizes.
HMSGD with fixed step-size	It is sensitive to tuning and often leads to erratic estimate trajectories.	HMSGD with Adam/1D sampling typically works well with default tuning parameters and estimate trajectories are less susceptible to erratic jumps.
MCNR	Approximate Hessian is required and can be ill-conditioned. It might be sensitive to initial values.	HMSGD with Adam/1D sampling does not require Hessian and is less sensitive to initial values.

We summarize the challenges of existing MLE methods and their solutions via HMSGD-based methods in Table 1. In general, gradient-descent-based approaches might not be able to find the global optimum if the objective function is nonconvex or close to being flat in certain regions. Since HMSGD is based on gradient descent, we also expect HMSGD to break down for nonconvex or almost flat likelihood functions.

There have been recent approaches to improving and understanding the performance of adaptive step-size algorithms like Adam, including approaches to tuning the step size [34,45,73]. Thanks to the unifying framework of this paper, those advances can be straightforwardly applied to the likelihood estimation for general hierarchical model setting. We provide access to these algorithms in an easy-to-use implementation that is still customizable, and we hope these techniques can be valuable to practitioners in many fields and speed up computations that would otherwise be infeasible.

Notes

1. The number of samples used in the gradient approximation or the 1D sampling is only 150, since the first half of the samples are discarded. Similar comments also apply for a different choice of sample size.
2. We use `optim()` in R.

Acknowledgments

We thank Chris Paciorek for his numerous suggestions in the project, in particular the idea of warm start in the MCMC sampling. We also thank Nick Michaud for NIMBLE's implementation of MCEM. Finally, we thank the anonymous reviewers for their insightful comments.

Data availability statement

The datasets used in the numerical experiments are all previously published. The data used in the *pump* and *seeds* examples come from various NIMBLE documentations [52,53]. The data used in the *salamander* example come from the glmm R package [42]. The data used in the multi-species occurrence model come from a study by Zipkin et al. [76]. All code used to run these examples can be found in the GitHub repository: https://github.com/jcyhong/MCMCmaxlik-dev/.

Disclosure statement

The authors declare that they have no conflict of interest. This paper is based on a chapter of the first author's dissertation available online as a working paper: https://escholarship.org/content/qt3x47n13t/qt3x47n13t_noSplash_6ad0bbdd7ff92f1587f4d4c690e154fe.pdf

Funding

The first author is funded in part by National Science Foundation through Research Training Groups (RTG) Award #1745640. The second author was funded in part by the Gordon and Betty Moore Foundation through Grant GBMF3834 and by the Alfred P. Sloan Foundation through Grant 2013-10-27 to the University of California, Berkeley and in part by the National Physical Science Consortium.

ORCID

Sara Stoudt http://orcid.org/0000-0002-1693-8058

References

[1] M.A. Abba, B.J. Reich, R. Majumder, and B. Feng, *Stochastic gradient mcmc for massive geostatistical data*, preprint (2024), arXiv:2405.04531.
[2] S. Asmussen and P.W. Glynn, *Stochastic Simulation: Algorithms and Analysis*, Stochastic Modelling and Applied Probability, Springer, 2007.
[3] C. Baey, M. Delattre, E. Kuhn, J.B. Leger, and S. Lemler, *Efficient preconditioned stochastic gradient descent for estimation in latent variable models*, in *International Conference on Machine Learning*, PMLR, 2023, pp. 1430–1453.
[4] D. Bates, M. Maechler, B. Bolker, and S. Walker, *Fitting linear mixed-effects models using lme4*, J. Stat. Softw. 67 (2015), pp. 1–48.
[5] M.J. Beal and Z. Ghahramani, *The variational Bayesian EM algorithm for incomplete data: With application to scoring graphiclal model structures*, Bayesian Anal. 7 (2003), pp. 453–463.
[6] L. Bottou (ed.), *Stochastic Gradient Learning in Neural Networks*, Neuro-Nimes, Holmdel, NJ, 1991.
[7] L. Bottou (ed.), *Large-Scale Machine Learning with Stochastic Gradient Descent*, COMPSTAT, Paris, 2010.
[8] S. Brooks, A. Gelman, G.L. Jones, and X. Meng, *Handbook of Markov Chain Monte Carlo*, Chapman and Hall/CRC, Boca Raton, FL, 2011.
[9] B.S. Caffo, W. Jank, and G. Jones, *Ascent-based Monte Carlo expectation maximization*, J. R. Stat. Soc., B: Stat. Methodol. 67 (2005), pp. 235–251.
[10] O. Cappé, E. Moulines, and T. Ryden, *Inference in Hidden Markov Models, 1*, Springer-Verlag, New York, 2005.
[11] J.F. Cardoso, M. Lavielle, and E. Moulines, *Un algorithme d'identification par maximum de vraisemblance pour des données incomplètes*, C.R. Acad. Sci. Paris Série I Statistique 320 (1995), pp. 363–368.
[12] G. Celeux and J. Diebolt, *The SEM algorithm: A probabilistic teacher algorithm derived from the EM algorithm for the mixture problem.*, Comput. Stat. 2 (1985), pp. 73–82.
[13] G. Celeux and J. Diebolt, *Une version de type recuit simulé de l'algorithme EM.*, C. R. Acad.Sci. Paris Sér. I Math. 310 (1990), pp. 119–124.
[14] S. Chib and I. Jeliazkov, *Marginal likelihood from the metropolis–hastings output*, J. Am. Stat. Assoc. 96 (2001), pp. 270–281.
[15] J. Clark, *Why environmental scientists are becoming Bayesians.*, Ecol. Lett. 8 (2005), pp. 2–14.
[16] N. Cressie, C.A. Calder, J. Clark, J. Hoef, and C. Wikle, *Accounting for uncertainty in ecological analysis: The strengths and limitations of hierarchical statistical modeling.*, Ecol. Appl. 19 (2009), pp. 553–570.

[17] M.J. Crowder, *Beta-binomial Anova for proportions*, J. R. Stat. Soc., C: Appl. Stat. 27 (1978), pp. 34–37.
[18] P. Das and S. Ghosal, *Bayesian non-parametric simultaneous quantile regression for complete and grid data*, Comput. Stat. Data Anal. 127 (2018), pp. 172–186.
[19] P. de Valpine, *Monte Carlo state-space likelihoods by weighted posterior kernel density estimation*, J. Am. Stat. Assoc. 99 (2004), pp. 523–536.
[20] P. de Valpine, *Frequentist analysis of hierarchical models for population dynamics and demographic data*, J. Ornithol. 152 (2012), pp. 393–408.
[21] P. de Valpine, C. Paciorek, D. Turek, N. Michaud, C. Anderson-Bergman, F. Obermeyer, C. Wehrhahn Cortes, A. Rodriguez, D. Temple Lang, and S. Paganin, *NIMBLE: MCMC, Particle Filtering, and Programmable Hierarchical Modeling*, 2020. R package version 0.11.0, Available at https://cran.r-project.org/package=nimble.
[22] P. de Valpine, D. Turek, C.J. Paciorek, C. Anderson-Bergman, D.T. Lang, and R. Bodik, *Programming with models: Writing statistical algorithms for general model structures with nimble*, J. Comput. Graph. Stat. 26(2) (2017), pp. 403–413.
[23] B. Delyon, M. Lavielle, and E. Moulines, *Convergence of a stochastic approximation version of the EM algorithm*, Ann. Stat. 27 (1999), pp. 94–128.
[24] A.P. Dempster, N.M. Laird, and D.B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, J. R. Stat. Soc., B: Methodol. 39 (1977), pp. 1–22.
[25] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online leaning and stochastic optimization*, J. Mach. Learn. Res. 12 (2011), pp. 2121–2159.
[26] B. Efron, *Discussion on maximum likelihood from incomplete data via the EM algorithm*, J. R. Stat. Soc., B: Methodol. 39 (1977), pp. 1–38.
[27] B. Efron, *Bayesians, frequentists, and scientists*, J. Am. Stat. Assoc. 100 (2005), pp. 1–5.
[28] D.P. Gaver and I.G. O'Muircheartaigh, *Robust empirical Bayes analyses of event rates*, Technometrics 29 (1987), pp. 1–15.
[29] A. Gelman and D.B. Rubin, *Inference from iterative simulation using multiple sequences*, Neural Comput. 7 (1992), pp. 457–472.
[30] E.I. George, U.E. Makov, and A.F.M. Smith, *Conjugate likelihood distributions*, Scand. J. Stat. 20 (1993), pp. 147–156.
[31] G. Hinton and R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, Science 313 (2006), pp. 504–507.
[32] G.E. Hinton, S. Osindero, and Y. Teh, *A fast learning algorithm for deep belief networks*, Neural Comput. 18 (2006), pp. 1527–1554.
[33] M. Hooten and N. Hobbs, *A guide to Bayesian model selection for ecologists*, Ecol. Monogr. 85 (2015), pp. 3–28.
[34] F. Huang, J. Li, and H. Huang, *Super-adam: Faster and universal framework of adaptive gradients*, Adv. Neural Inf. Process. Syst. 34 (2021), pp. 9074–9085.
[35] S. Jackman, *Bayesian Analysis for the Social Sciences*, Vol. 846, John Wiley and Sons, Chippenham, 2009.
[36] E. Jacquier, M. Johannes, and N. Polson, *MCMC maximum likelihood for latent state models*, J. Econom. 137 (2007), pp. 615–640.
[37] M.R. Karim and S.L. Zeger, *Generalized linear models with random effects; salamander mating revisited*, Biometrics 48 (1992), pp. 631–644.
[38] B. Karimi, M. Lavielle, and E. Moulines, *f-saem: A fast stochastic approximation of the em algorithm for nonlinear mixed effects models*, 2018. hal-01958248.
[39] R.L. Kashyap and C.C. Blaydon, *Estimation of probability density and distribution functions*, IEEE Trans. Inf. Theory 14(4) (1968), pp. 549–556.
[40] J. Kiefer and J. Wolfowitz, *Stochastic estimation of the maximum of a regression function*, Ann. Math. Stat. 23(3) (1952), pp. 462–466.
[41] D.P. Kingma and J.L. Ba, *Adam: A method for stochastic optimization*, in *ICLR 2015*, San Diego, CA, 2015.
[42] C. Knudson, *glmm: Generalized Linear Mixed Models via Monte Carlo likelihood approximation*, 2016. Available at https://CRAN.R-project.org/package=glmm.

[43] E. Kuhn and M. Lavielle, *Coupling a stochastic approximation version of EM with an MCMC procedure*, ESAIM – Probab. Stat. 8 (2004), pp. 115–131.
[44] A.Y.C. Kuk and Y.W. Cheng, *The Monte Carlo Newton-Raphson algorithm*, J. Stat. Comput. Simul. 59 (1997), pp. 233–250.
[45] N. Landro, I. Gallo, and R. La Grassa, *Mixing adam and sgd: A combined optimization method*, preprint (2020), arXiv:2011.08042.
[46] A. Lawson, *Bayesian Disease Mapping: Hierarchical Modeling in Spatial Epidemiology*, CRC Press, Boca Raton, FL, 2013.
[47] S.R. Lele, B. Dennis, and F. Lutscher, *Data cloning: Easy maximum likelihood estimation for complex ecological models using bayesian markov chain monte carlo methods*, Ecol. Lett. 10(7) (2007), pp. 551–563.
[48] Z. Liu, *A sequential monte carlo gibbs coupled with stochastically approximated expectation-maximization algorithm for functional data*, Stat. Interface, 15 (2022), pp. 197–208.
[49] T.A. Louis, *Finding the observed information matrix when using the EM algorithm*, J. R. Stat. Soc., B: Methodol. 44 (1982), pp. 226–233.
[50] D. Lunn, C. Jackson, N. Best, A. Thomas, and D. Spiegelhalter, *The BUGS Book: A Practical Introduction to Bayesian Analysis*, CRC Press, New York, NY, 2012.
[51] C.E. McCulloch, *Maximum likelihood algorithms for generalized linear mixed models*, J. Am. Stat. Assoc. 92 (1997), pp. 162–170.
[52] NIMBLE Team, *Creating and running a Markov chain Monte Carlo (MCMC) algorithm in NIMBLE* (ND). Available at https://r-nimble.org/nimbleExamples/nimble_basic_mcmc.html.
[53] NIMBLE Team, *MCMC for logistic regression with random effects* (ND). Available at https://r-nimble.org/nimbleExamples/nimble_logistic_regression.html.
[54] M. Parzen, S. Ghosh, S. Lipsitz, D. Sinha, G.M. Fitzmaurice, B.K. Mallick, and J.G. Ibrahim, *A generalized linear mixed model for longitudinal binary data with a marginal logit link function.*, Ann. Appl. Stat. 5 (2011), pp. 449–467.
[55] B.A. Pearlmutter, *Fast exact multiplication by the hessian*, Neural Comput. 6(1) (1993), pp. 147–160.
[56] U. Picchini and A. Samson, *Coupling stochastic EM and approximate bayesian computation for parameter inference in state-space models*, Comput. Stat. 33 (2018), pp. 179–212.
[57] A. Plakhov and P. Cruz, *A stochastic approximation algorithm with step-size adaptation*, J. Math. Sci. 120 (2004), pp. 964–973.
[58] L.C. Ponisio, P.d. Valpine, N. Michaud, and D. Turek, *One size does not fit all: Customizing MCMC methods for hierarchical models using NIMBLE*, Ecol. Evol. 10 (2020), pp. 2385–2416. https://doi.org/10.1002/ece3.6053.
[59] A. Rakhlin, O. Shamir, and K. Sridharan, *Making gradient descent optimal for strongly convex stochastic optimization*, in *International Conference on Machine Learning*, Edinburgh, 2012.
[60] H. Robbins and S. Monro, *A stochastic approximation method*, Ann. Math. Stat. 22 (1951), pp. 400–407.
[61] T.M.K. Roeder, P.I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S.E. Chick (eds.), *Warm starting Bayesian optimization*, in *Winter Simulation Conference*, Arlington, VA, 2016.
[62] A. Royle and R. Dorazio, *Hierarchical Modeling and Inference in Ecology: The Analysis of Data From Populations, Metapopulations and Communities.*, Academic Press, San Diego, CA, 2008.
[63] H. Rue, S. Martino, and N. Chopin, *Approximate Bayesian inference for latent Gaussian models using integrated nested laplace approximations*, J. R. Stat. Soc., B: Stat. Methodol. 71 (2008), pp. 319–392.
[64] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, Boca Raton, FL, 1986.
[65] J.C. Spall, *A stochastic approximation technique for generating maximum likelihood parameter estimates*, in *Proceedings of American Control Conference*, Minneapolis, MN, 1987.
[66] J. Spall, *Multivariate stochastic approximation using a simultaneous perturbation gradient approximation.*, IEEE Trans. Autom. Control 37(3) (1992), pp. 332–341.
[67] M.J. Wainwright and M.I. Jordan, *Graphical models, exponential families, and variational inference*, Found. Trends Mach. Learn. 1-2 (2008), pp. 1–305.

[68] A. Wald and J. Wolfowitz, *On a test whether two samples are from the same population*, Ann. Math. Stat. 11 (1940), pp. 147–162.
[69] M.P. Wand and M.C. Jones, *Kernel Smoothing*, Chapman and Hall/CRC, New York, NY, 1995.
[70] K. Wang, T. Bui-Thanh, and O. Ghattas, *A randomized maximum a posterior method for posterior sampling of high dimensional nonlinear Bayesian inverse problems*, SIAM J. Sci. Comput. 40 (2018), pp. A142–A171.
[71] G.C. Wei and M.A. Tanner, *A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms*, J. Am. Stat. Assoc. 85 (1990), pp. 699–704.
[72] P. Wolfe, *Convergence conditions for ascent methods*, SIAM Rev. 11 (1969), pp. 226–235.
[73] W. Yuan and K.X. Gao, *Eadam optimizer: How ϵ impact adam*, preprint (2020), arXiv:2011.02150.
[74] M.D. Zeiler, *Adadelta: An adaptive learning method*, 2021, arXiv:1212.5701.
[75] B. Zhou, L. Gao, and Y. Dai, *Gradient methods with adaptive step-sizes*, Comput. Optim. Appl. 35 (2006), pp. 69–86.
[76] E.F. Zipkin, J.A. Royle, D.K. Dawson, and S. Bates, *Multi-species occurrence models to evaluate the effects of conservation and management actions*, Biol. Conserv. 143 (2010), pp. 479–484.

Appendix. Numerical results

In the following tables, Exec.(s) refers to the CPU time for 300 iterations in terms of seconds; Conv.(s) refers to the CPU time to convergence in terms of seconds; Conv.(iter.) refers to the number of iterations to convergence; loglik diff. refers to the log likelihood difference between the resulting estimate and the benchmark estimate; NA in Conv. means that the convergence test is not passed within 300 iterations.

For each of the case studies, we also randomly initialize 30 parameter values based on certain distributions (see the captions of the respective tables for the exact distributions), and compute the root-mean-squared deviation (RMSE) between the resulting estimates and the benchmark estimate.

In all of the examples, for MCEM, the MCMC sample sizes are not fixed; they are adaptively increased as described in [9] for convergence reasons.

Table A1. Numerical results for the *pump* example with MCMC sample size **300** and initial value (**10**, **10**). The benchmark MLE (0.823, 1.262) can be obtained numerically.

	α	β	Exec.(s)	Conv.(s)	Conv.(iter.)	loglik diff.
Fixed (0.05)	0.858	1.360	1.612	NA	NA	0.00731
Fixed (0.005)	4.676	11.251	1.551	NA	NA	9.91000
Adam	0.821	1.242	1.521	0.817	162	0.00049
Newton-Raphson	0.825	1.270	2.167	0.439	59	0.00005
1D Sampling	0.954	1.530	6.526	1.245	57	0.06283
MCEM	0.834	1.311	0.368	0.368	5	0.00216

Table A2. Numerical results for the *pump* example with MCMC sample size **300** and initial value (**10**, **2**). The benchmark MLE (0.823, 1.262) can be obtained numerically.

	α	β	Exec.(s)	Conv.(s)	Conv.(iter.)	loglik diff.
Fixed (0.05)	0.821	1.263	1.658	NA	NA	0.00007
Fixed (0.005)	2.505	6.236	1.542	NA	NA	4.50666
Adam	0.826	1.266	1.519	1.237	245	0.00005
Newton-Raphson	0.822	1.252	2.141	0.344	47	0.00010
1D Sampling	0.828	1.191	6.602	1.085	47	0.01253
MCEM	0.821	1.257	0.640	0.640	10	0.00002

Table A3. Numerical results for the *pump* example with MCMC sample size **3000** and initial value (**10**, **10**). The benchmark MLE (0.823, 1.262) can be obtained numerically.

	α	β	*Exec.*(*s*)	Conv.(s)	Conv.(iter.)	loglik diff.
Fixed (0.05)	0.863	1.374	7.708	NA	NA	0.00941
Fixed (0.005)	4.677	11.253	7.230	NA	NA	9.91292
Adam	0.822	1.264	7.784	5.410	212	0.00002
Newton-Raphson	0.824	1.262	12.811	2.466	58	0.00001
1D Sampling	0.820	1.254	21.061	5.719	84	0.00005
MCEM	0.834	1.311	0.368	0.368	5	0.00216

Table A4. Numerical results for the *pump* example with MCMC sample size **3000** and initial value (**10**, **2**). The benchmark MLE (0.823, 1.262) can be obtained numerically.

	α	β	*Exec.*(*s*)	Conv.(s)	Conv.(iter.)	loglik diff.
Fixed (0.05)	0.823	1.262	7.835	7.199	276	0.00000
Fixed (0.005)	2.505	6.236	7.386	NA	NA	4.50628
Adam	0.824	1.261	7.664	NA	NA	0.00002
Newton-Raphson	0.822	1.256	12.634	2.152	51	0.00003
1D Sampling	0.826	1.263	20.193	9.416	138	0.00007
MCEM	0.821	1.257	0.640	0.640	10	0.00002

Table A5. Root-mean-squared errors (RMSEs) for the *pump* example with MCMC sample size **300** (except for MCEM, which chooses the MCMC sample sizes adaptively) with 30 random initializations based on $(\alpha^{(0)}, \beta^{(0)}) \sim \text{Unif}[0.05, 15] \times [0.05, 15]$.

	α	β
Fixed (0.05)	0.365	1.16
Fixed (0.005)	3.38	8.52
Adam	0.0768	0.238
Newton-Raphson	0.00340	0.221
1D Sampling	0.0499	0.104
MCEM	0.00468	0.0126

Note: The benchmark MLE (0.823, 1.262) can be obtained numerically.

Table A6. Root-mean-squared errors (RMSEs) for the *pump* example with MCMC sample size **3000** (except for MCEM, which chooses the MCMC sample sizes adaptively) with 30 random initializations based on $(\alpha^{(0)}, \beta^{(0)}) \sim \text{Unif}[0.05, 15] \times [0.05, 15]$.

	α	β
Fixed (0.05)	0.474	1.58
Fixed (0.005)	3.27	7.73
Adam	0.00419	0.0156
Newton-Raphson	0.000876	0.00315
1D Sampling	0.0158	0.0399
MCEM	0.00468	0.0126

Note: The benchmark MLE (0.823, 1.262) can be obtained numerically.

Table A7. Numerical results for the *seeds* example with MCMC sample size **20** and initial value (**0**, **0**, **1**). The *lme4* estimate (in bold) should be viewed as the benchmark for the MLE estimate. The glmer package relies heavily on the Gaussian assumption for random effects and leverages special case computations to drastically reduce the computational time.

	β_0	β_1	σ_{RE}	Exec.(s)	Conv.(s)	Conv. (iter.)	*loglikdiff*.
Fixed (0.05)	0.183	1.709	3.077	0.683	NA	NA	15.63014
Fixed (0.005)	−0.527	1.332	0.297	0.800	0.419	146	0.09083
Adam	−0.516	1.342	0.385	0.706	0.534	231	0.48930
Newton-Raphson	−0.552	1.296	0.060	0.806	0.184	58	0.88618
1D Sampling	−0.526	1.328	0.257	5.949	NA	NA	0.03303
MCEM	−0.547	1.309	0.253	5.540	5.540	33	0.00051
glmer (lme4)	**−0.519**	**1.019**	**0.307**	0.100	0.100	NA	0.00000

Table A8. Numerical results for the *seeds* example with MCMC sample size **20** and initial value (**−1**, **−1**, **4**). The *lme4* estimate (in bold) should be viewed as the benchmark for the MLE estimate.

	β_0	β_1	σ_{RE}	Exec.(s)	Conv.(s)	Conv. (iter.)	loglik diff.
Fixed (0.05)	0.036	2.687	9.477	0.637	NA	NA	26.41578
Fixed (0.005)	−0.601	1.382	0.306	0.738	NA	NA	0.13226
Adam	−0.495	1.207	0.297	0.742	0.705	283	0.15175
Newton-Raphson	−1.031	−0.931	10.000	0.902	NA	NA	27.08173
1D Sampling	−0.590	1.388	0.326	6.050	1.700	81	0.19080
MCEM	−0.550	1.315	0.252	3.888	3.888	53	0.00038
glmer (lme4)	**−0.519**	**1.019**	**0.307**	0.100	0.100	NA	0.00000

Note: The glmer package relies heavily on the Gaussian assumption for random effects and leverages special case computations to drastically reduce the computational time.

Table A9. Numerical results for the *seeds* example with MCMC sample size **300** and initial value (**0**, **0**, **1**). The *lme4* estimate (in bold) should be viewed as the benchmark for the MLE estimate.

	β_0	β_1	σ_{RE}	Exec.(s)	Conv.(s)	Conv. (iter.)	loglik diff.
Fixed (0.05)	−0.530	1.367	0.572	1.239	0.232	58	1.87545
Fixed (0.005)	−0.545	1.312	0.261	1.367	0.968	208	0.00493
Adam	−0.541	1.294	0.246	1.268	0.529	118	0.00255
Newton-Raphson	−0.547	1.304	0.248	1.776	0.929	148	0.00040
1D Sampling	−0.539	1.313	0.264	7.201	5.596	233	0.00937
MCEM	−0.547	1.309	0.253	5.540	5.540	33	0.00051
glmer (lme4)	**−0.519**	**1.019**	**0.307**	0.100	0.100	NA	0.00000

Note: The glmer package relies heavily on the Gaussian assumption for random effects and leverages special case computations to drastically reduce the computational time.

Table A10. Numerical results for the *seeds* example with MCMC sample size **300** and initial value (**−1**, **−1**, **4**). The *lme4* estimate (in bold) should be viewed as the benchmark for the MLE estimate.

	β_0	β_1	σ_{RE}	Exec.(s)	Conv.(s)	Conv. (iter.)	loglik diff.
Fixed (0.05)	−1.946	0.916	3.838	1.287	NA	NA	18.39451
Fixed (0.005)	−0.532	1.275	0.248	1.308	NA	NA	0.01063
Adam	−0.552	1.289	0.258	1.217	0.271	64	0.01439
Newton-Raphson	−0.996	−1.000	10.000	1.875	NA	NA	27.08599
1D Sampling	−0.539	1.300	0.271	6.895	4.490	196	0.01645
MCEM	−0.550	1.315	0.252	3.888	3.888	53	0.00038
glmer (lme4)	**−0.519**	**1.019**	**0.307**	0.100	0.100	NA	0.00000

Note: The glmer package relies heavily on the Gaussian assumption for random effects and leverages special case computations to drastically reduce the computational time.

Table A11. Root-mean-squared errors (RMSEs) for the *seeds* example with MCMC sample size **20** (except for MCEM, which chooses the MCMC sample sizes adaptively) with 30 random initializations drawn independently from $\beta_0 \sim$ Uniform$[-10, 10]$, $\beta_1 \sim$ Uniform$[-10, 10]$, and $\sigma_{RE} \sim$ Uniform$[0.05, 15]$.

	β_0	β_1	σ_{RE}
Fixed (0.05)	1.27	1.30	4.46
Fixed (0.005)	4.26	5.20	6.62
Adam	0.904	1.09	1.53
1D Sampling	0.566	1.50	0.782
MCEM	4.53	4.25	5.83

Note: The benchmark MLE is based on the *lme4* estimate. Newton-Raphson diverges for certain initializations and hence the corresponding RMSEs are omitted.

Table A12. Root-mean-squared errors (RMSEs) for the *seeds* example with MCMC sample size **300** (except for MCEM, which chooses the MCMC sample sizes adaptively) with 30 random initializations drawn independently from $\beta_0 \sim$ Uniform$[-10, 10]$, $\beta_1 \sim$ Uniform$[-10, 10]$, and $\sigma_{RE} \sim$ Uniform$[0.05, 15]$.

	β_0	β_1	σ_{RE}
Fixed (0.05)	1.97	1.45	4.08
Fixed (0.005)	3.41	4.76	6.52
Adam	0.0333	0.301	0.441
1D Sampling	0.0327	0.290	0.0445
MCEM	4.53	4.25	5.83

Note: The benchmark MLE is based on the *lme4* estimate. Newton-Raphson diverges for certain initializations and hence the corresponding RMSEs are omitted.

Table A13. Numerical results for the *salamander* example with MCMC sample size **300** and initial value (**2**, **2**, **2**, **2**, **2**, **2**).

	β_1	β_2	β_3	β_4	σ_F^2	σ_M^2	Exec.(s)	loglik diff.
Fixed (0.05)	1.006	0.310	−1.933	0.992	1.358	1.211	15.036	0.09266
Fixed (0.005)	1.078	0.355	−2.007	1.044	1.634	1.412	14.626	0.38769
Adam	1.052	0.332	−1.916	1.013	1.417	1.338	14.636	0.20656
Newton-Raphson	1.028	0.320	−1.946	0.979	1.406	1.207	26.113	0.11465
1D Sampling	1.089	0.314	−1.951	0.977	1.539	1.250	71.891	0.22921
MCEM	1.011	0.325	−1.957	1.026	1.201	1.130	14.095	0.16727
glmm	1.023	0.335	−1.908	1.006	1.326	1.221	1181.430	0.08856
glmer (lme4)	**1.008**	**0.306**	**−1.896**	**0.990**	**1.174**	**1.041**	0.100	0.00000

Note: The *lme4* estimates (in bold) should be viewed as the benchmark for the MLE estimate. Fixed step-sizes, Adam, Newton-Raphson, and 1D sampling did not pass the convergence criterion within 300 iterations.

Table A14. Numerical results for the *salamander* example with MCMC sample size **300** and initial value (**4**, **4**, **4**, **4**, **4**, **4**).

	β_1	β_2	β_3	β_4	σ_F^2	σ_M^2	Exec.(s)	loglik diff.
Fixed (0.05)	1.036	0.330	−1.932	1.005	1.392	1.257	14.719	0.12994
Fixed (0.005)	1.748	0.835	−2.041	1.722	3.754	3.545	14.938	6.59630
Adam	1.008	0.291	−1.946	0.968	1.343	1.301	14.569	0.15004
Newton-Raphson	1.005	0.288	−1.911	0.972	1.265	1.260	26.109	0.10397
1D Sampling	1.125	0.385	−2.021	1.053	1.625	1.578	73.098	0.54268
MCEM	1.024	0.330	−1.935	0.996	1.178	1.110	23.432	0.11645
glmm	1.023	0.335	−1.908	1.006	1.326	1.221	1181.430	0.08856
glmer (lme4)	**1.008**	**0.306**	**−1.896**	**0.990**	**1.174**	**1.041**	0.100	0.00000

Note: The *lme4* estimates (in bold) should be viewed as the benchmark for the MLE estimate. Fixed step-sizes, Adam, Newton-Raphson, and 1D sampling did not pass the convergence criterion within 300 iterations.

Table A15. Root-mean-squared errors (RMSEs) for the *salamander* example with MCMC sample size **300** (except for MCEM, which chooses the MCMC sample sizes adaptively) with 30 random initializations drawn independently from $\beta_1 \sim$ Uniform[−10, 10], $\beta_2 \sim$ Uniform[−10, 10], $\beta_3 \sim$ Uniform[−10, 10], $\beta_4 \sim$ Uniform[−10, 10], $\sigma_F^2 \sim$ Uniform[0.05, 15], and $\sigma_M^2 \sim$ Uniform[0.05, 15].

	β_1	β_2	β_3	β_4	σ_F^2	σ_M^2
Fixed (0.05)	0.0276	0.0219	0.0571	0.0220	0.236	0.218
Fixed (0.005)	1.19	1.12	1.50	1.30	7.44	6.80
Adam	0.0535	0.0375	0.119	0.0563	0.356	0.620
1D Sampling	0.0436	0.0398	0.0812	0.0416	0.328	0.283
MCEM	0.0184	0.0244	0.0587	0.0190	0.224	0.218

Note: The benchmark MLE is based on the *lme4* estimate. Newton-Raphson diverges for certain initializations and hence the corresponding RMSEs are omitted.

Table A16. Numerical results (mean components only) for the multi-species occurrence example with MCMC sample size **300** and initial value 0 for each of the parameters.

	μ_{uCATO}	μ_{uFCW}	μ_{vCATO}	μ_{vFCW}	μ_{a1}	μ_{a2}	μ_{a3}	μ_{a4}	μ_{b1}	μ_{b2}
Fixed (0.05)	0.432	0.58	0.175	0.244	0.991	0.52	−0.198	1.743	−13.749	15.585
Fixed (0.005)	0.319	0.442	0.197	0.259	0.427	0.014	−0.204	0.131	−0.139	0.01
Adam	0.412	0.506	0.17	0.227	0.533	−0.128	−0.157	0.143	−0.123	0.096
Newton Raphson	1.000	0.067	0.004	1.000	1.294	−1.297	−0.167	1.83	−0.811	12.563
MCEM	**0.353**	**0.465**	**0.186**	**0.248**	**0.457**	**0.005**	**−0.154**	**0.127**	**−0.133**	**0.086**

Note: The *MCEM* estimates (in bold) should be viewed as the benchmark for the MLE estimate. Fixed step-sizes, Adam, and Newton-Raphson did not pass the convergence criterion within 300 iterations. 1D sampling ran into trouble for MCMC sampling and failed to yield a parameter estimate.

Table A17. Numerical results (variance components only) for the multi-species occurrence example with MCMC sample size **300** and initial value 0 for each of the parameters.

	σ_{uCATO}	σ_{uFCW}	σ_{vCATO}	σ_{vFCW}	σ_{a1}	σ_{a2}	σ_{a3}	σ_{a4}	σ_{b1}	σ_{b2}
Fixed (0.05)	5.131	5.042	1.509	1.405	1.208	1.369	0.473	2.584	11656.74	2194570.04
Fixed (0.005)	3.122	2.725	1.307	1.179	0.622	0.14	0.092	0.23	0.23	0.306
Adam	3.589	2.918	1.376	1.276	0.666	0.032	0.033	0.259	0.188	0.03
Newton Raphson	29.91	2.137	1.918	56.644	0.867	1.038	0.374	0.029	4.496	0.003
MCEM	3.259	2.728	1.328	1.192	0.637	0.243	0.171	0.251	0.2	0.048

Note: The *MCEM* estimates (in bold) should be viewed as the benchmark for the MLE estimate. Fixed step-sizes, Adam, and Newton-Raphson did not pass the convergence criterion within 300 iterations. 1D sampling ran into trouble for MCMC sampling and failed to yield a parameter estimate.

Table A18. Execution time for the multi-species occurrence example with MCMC sample size **300** and initial value 0 for each of the parameters.

	Exec.(s)
Fixed (0.05)	1410.891
Fixed (0.005)	1382.729
Adam	1396.477
Newton-Raphson	1416.499
MCEM	9859.548

Table A19. Root-mean-squared errors (RMSEs) for the multi-species occurrence example (mean components only) with MCMC sample size **300** with 30 random initializations drawn independently from Uniform($[-1, 1]^{20}$).

	μ_{uCATO}	μ_{uFCW}	μ_{vCATO}	μ_{vFCW}	μ_{a1}	μ_{a2}	μ_{a3}	μ_{a4}	μ_{b1}	μ_{b2}
Fixed (0.05)	0.333	0.21	0.082	0.046	1.594	0.4	4.816	4.69	18.633	23.598
Fixed (0.005)	0.477	0.161	0.099	0.086	1.59	7.374	0.117	20.569	0.007	0.232
Adam	0.522	0.351	0.083	0.09	2.865	3.077	2.062	1.9	0.009	0.022

Note: The benchmark MLE is based on the *MCEM* estimate. Newton-Raphson diverges for certain initializations; 1D sampling frequently encounters MCMC issues; MCEM often reaches non-finite log-likelihood in its iterations and fails to continue. Hence their corresponding RMSEs are omitted.

Table A20. Root-mean-squared errors (RMSEs) for the multi-species occurrence example (variance components only) with MCMC sample size **300** with 30 random initializations drawn independently from Uniform($[-1, 1]^{20}$).

	σ_{uCATO}	σ_{uFCW}	σ_{vCATO}	σ_{vFCW}	σ_{a1}	σ_{a2}	σ_{a3}	σ_{a4}	σ_{b1}	σ_{b2}
Fixed (0.05)	4.5×10^9	1.9×10^6	1.6×10^4	0.826	3.7×10^6	7.0×10^6	0.292	2.2×10^9	3.0×10^7	9.0×10^7
Fixed (0.005)	454.217	229.01	0.564	0.38	117.251	169.306	0.291	263.85	0.037	0.438
Adam	19.31	13.806	0.386	0.305	4.85	5.505	1.857	9.229	0.019	0.022

Note: The benchmark MLE is based on the *MCEM* estimate. Newton-Raphson diverges for certain initializations; 1D sampling frequently encounters MCMC issues; MCEM often reaches non-finite log-likelihood in its iterations and fails to continue. Hence their corresponding RMSEs are omitted.