Hamiltonian Monte Carlo Inference of Marginalized Linear Mixed-Effects Models

Jinlin Lai, Justin Domke, Daniel Sheldon
Manning College of Information and Computer Sciences
University of Massachusetts Amherst
{jinlinlai,domke,sheldon}@cs.umass.edu

Abstract

Bayesian reasoning in linear mixed-effects models (LMMs) is challenging and often requires advanced sampling techniques like Markov chain Monte Carlo (MCMC). A common approach is to write the model in a probabilistic programming language and then sample via Hamiltonian Monte Carlo (HMC). However, there are many ways a user can transform a model that make inference more or less efficient. In particular, marginalizing some variables can greatly improve inference but is difficult for users to do manually. We develop an algorithm to easily marginalize random effects in LMMs. A naive approach introduces cubic time operations within an inference algorithm like HMC, but we reduce the running time to linear using fast linear algebra techniques. We show that marginalization is always beneficial when applicable and highlight improvements in various models, especially ones from cognitive sciences[1].

1 Introduction

Bayesian hierarchical models account for complicated relationships in data by introducing hierarchical structures [23]. Among hierarchical models, linear mixed effects models (LMMs) are widely used in various scientific disciplines, including ecology [31], medicine [7], psychology [41], neuroscience [77] and cognitive science [47]. Solving LMMs involves inferring latent variables, such as fixed and random effects, based on the observed data. Fixed effects are shared by all observations, while random effects vary across different groups within the data. LMMs are often implemented using probabilistic programming languages (PPLs), which isolate inference from modeling: users write a program representing the model and the PPL automatically executes a suitable inference algorithm. Variants of Hamiltonian Monte Carlo (HMC) [15] are dominant in many PPLs today and are widely used for LMMs. For example, BRMS [8] is an influential R package that allows users to write regression-style formulas that are automatically translated to Stan programs [9] representing an LMM, and then Stan's HMC implementation is called to generate posterior samples.

We develop techniques that allow users to easily transform their models to analytically marginalize random effect variables from LMMs to improve the efficiency of HMC. Marginalization has several benefits. First, there are often pathologies in LMMs that hinder efficient HMC sampling. A notable one is the "funnel" shape created by correlation between variance parameters and parameters for fixed or random effects [45]. Marginalization [35] and other program transformations [26] have been shown to be useful in addressing such pathologies. Second, marginalization reduces the number H of latent variables for HMC. The complexity of HMC is about $O(H^{5/4})$ [11, 46], so it is desirable to run HMC on a subset of variables if marginalization can be done efficiently. Our methods enable marginalization of random effects in LMMs with a linear Gaussian structure, which includes models with normal and log-normal likelihoods as well as other likelihoods for continuous data based on

[1]The code is available at https://github.com/lll6924/hamiltonian_lme.git.

transforming a normal distribution. Note that our methods are not limited to HMC, and could be applied to many inference algorithms.

There are several challenges to efficient marginalization. The automatic marginalization algorithm of [35] can be applied to LMMs but is limited to scalar random variables, so it requires users to construct the LMM as a graphical model with separate variables for each effect and observation. Another alternative is to model the relationships between effects and observations with a design matrix and marginalize effects using properties of multivariate normal distributions. We call this the "vectorized approach" since it can leverage vectorization to accelerate computations. Unfortunately, vectorized marginalization leads to a dense covariance matrix over the observations and thus cubic time for evaluating the log-density within HMC, when the log-density of the original could be evaluated in linear time. Our main technical contribution is to accelerate vectorized marginalization for LMMs using fast linear algebra: we show that marginalization for a single random effect can be achieved with linear time complexity and can significantly accelerate HMC compared to both the original model and non-vectorized marginalization.

We implement vectorized marginalization for LMMs in NumPyro [5, 54] via simple classes users can use to express their models. We evaluate our approach on a variety of real LMMs from past scientific investigations, including nine models and datasets from cognitive sciences, and find that marginalization is always beneficial. Our findings suggest that practitioners should marginalize group-level effects whenever applicable in Bayesian hierarchical inference.

2 Background

To motivate our problem, we present an example model. In [72], a set of experiments were run to examine the relationship between human pupil and attention load. A total of $N = 2228$ measurements of pupil sizes from $M = 20$ subjects were taken under different attention load levels. Specifically, in the ith measurement, the pupil size $y_i \in \mathbb{R}^+$ of subject $g_i \in \{1, 2, ..., k\}$ under attention load $c_i \in \{0, 1, 2, 3, 4, 5\}$ was recorded. Pupil size can be assumed to have linear relationship $y_i \approx \theta_0 + \theta_1 c_i$ with respect to the attention load c_i, where both the slope θ_1 and intercept θ_0 split into fixed and random effects:

$$y_i = \alpha + u_{g_i,1} + c_i(\beta + u_{g_i,2}) + \epsilon, \ \ \epsilon \sim \mathcal{N}(0, \sigma^2),$$

where α, β are variables for fixed effects and $u_{\cdot,\cdot}$ are variables for subject-specific random effects. Bayesian hierarchical modeling assigns priors to each unknown variable:

$$\alpha \sim \mathcal{N}(1000, 500^2), \ \beta \sim \mathcal{N}(0, 100), \ \sigma \sim \mathcal{N}^+(0, 1000), \mathbf{T} \sim \mathcal{N}^+(\mathbf{0}, \text{diag}(1000^2, 1000^2)),$$
$$\mathbf{L}_u \sim \text{LKJCholesky}(2, 1), \ [u_{j,1}, u_{j,2}] \sim \mathcal{N}(\mathbf{0}, \mathbf{T}\mathbf{L}_u\mathbf{L}_u^T\mathbf{T}), \ j = 1, 2, ..., k.$$

A half-normal distribution ($\mathcal{N}^+$) and an LKJ distribution (LKJCholesky) [36] are used as a prior on the covariance matrix. Inference for the unknown parameters determining the relationship between pupil size and attention load can be performed by writing a probabilistic program and running HMC. For example, in NumPyro, the regression model for all measurements may be implemented as below.

```
numpyro.sample('y',dist.Normal(alpha+u[g][:,0]+c*(beta+u[g][:,1]),sigma),obs=y)
```

The code above uses advanced indexing and vectorization techniques in `numpy` where `u,g,c,y` are all vectors or matrices. We further observe that, conditioned on $\alpha, \beta, \sigma, \mathbf{T}, \mathbf{L}_u$, the distribution of all $\mathbf{u}_j$ and all y_i form a multivariate normal distribution. Theoretically it is possible to analytically integrate $\mathbf{u}$ out from the model to improve inference efficiency. But it is not straightforward for users to transform the probabilistic program to do so, and, as we will see, if done in the most obvious way, may not make the model more efficient for HMC.

To be more clear about how marginalization can be implemented, we rearrange the model into a canonical form that focuses on the random effects. All observations are collected into the vector $\mathbf{y} = [y_1, \cdots, y_N]^T$ and random effects into the vector $\mathbf{u} = [u_{1,1}, u_{1,2}, \cdots, u_{k,1}, u_{k,2}]^T$. Then, we can write

$$\mathbf{u} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_\mathbf{u}), \ \mathbf{y} \sim \mathcal{N}(\mathbf{A}\mathbf{u} + \mathbf{b}, \boldsymbol{\Sigma}_\mathbf{y}),$$

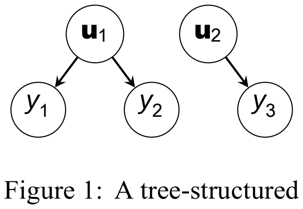

Figure 1: A tree-structured model conditioned on $\boldsymbol{\Theta}$.

where μ, $\mathbf{\Sigma_u}$, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{\Sigma_y}$ are functions of α, β, σ, $\mathbf{T}$, $\mathbf{L_u}$, g, c. Note that y_i only depends on the entry $\mathbf{u}_{g_i}$ of $\mathbf{u}$. The corresponding graphical model has a tree structure, as demonstrated in Figure 1. This tree structure has several benefits: first, matrix multiplications like $\mathbf{Au}$ and $\mathbf{A}^T\mathbf{y}$ can be done efficiently; second, we will see that it leads to a block-diagonal structure that facilitates efficient inversion in a key matrix that appears later.

For more general LMMs with more than one class of random effects we generalize the canonical form as

$$\mathbf{\Theta} \sim p(\mathbf{\Theta}), \quad \mathbf{u}_i|\mathbf{\Theta} \sim \mathcal{N}(\mu_i(\mathbf{\Theta}), \mathbf{\Sigma}_{\mathbf{u}_i}(\mathbf{\Theta})), \quad i = 1, 2, ..., L$$
$$\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_L \sim \mathcal{N}\left(\sum_{i=1}^{L} \mathbf{A}_i(\mathbf{\Theta})\mathbf{u}_i + \mathbf{b}(\mathbf{\Theta}), \mathbf{\Sigma_y}(\mathbf{\Theta})\right), \tag{1}$$

where $p(\mathbf{\Theta})$ is the distribution for global variables (including fixed effects), $p(\mathbf{u}_i|\mathbf{\Theta})$ is the distribution for random effects and $p(\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_1, ..., \mathbf{u}_L)$ is the distribution for observations. Notationally this generalization further adds an index to each random effect to specify its class. A user might specify the model directly in this canonical form, or in another syntax (e.g., the formula syntax of BRMS) that is compiled to this form. Each pair $(\mathbf{u}_i, \mathbf{A}_i)$ specifies a class of random effects for a particular classification of the observations (e.g., by subject, age, gender, etc.). Each classification contains multiple groups and different classifications are distinct from one another. Each observation belongs to one group for each classification. The vector $\mathbf{u}_i = [\mathbf{u}_{i,1}^T, \mathbf{u}_{i,2}^T, ..., \mathbf{u}_{i,k_i}^T]^T$ contains random effects for the ith classification (e.g., subject, age, or gender), consisting of k_i groups (e.g., one subject, age, or gender), with $\mathbf{u}_{i,j}$ containing the random effects (e.g., slope and intercept) for the jth group. We denote the number of observations as $\dim(\mathbf{y}) = N$, and the number of random effects per group as $\dim(\mathbf{u}_{i,j}) = d$. Any covariates—such as c_i in the pupil size example—are considered constants and not represented in the notation. In LMMs, the number d is related to the number of covariates and is usually small. The total number of random effects for $\mathbf{u}_i$ is denoted as $\dim(\mathbf{u}_i) = M_i = k_i d$. The matrix $\mathbf{A}_i$ therefore has size $N \times M_i$, and encodes the group structure for $\mathbf{u}_i$ by mapping random effects (together with covariates) to observations. Each row of $\mathbf{A}_i$ encodes the assignment of an observation to one group, so it has at most d nonzero elements. Therefore, the complexity of computing $\mathbf{A}_i\mathbf{u}_i$ is $O(Nd)$, as $\mathbf{A}$ has at most Nd nonzero elements. Henceforth, we omit the dependence on $\mathbf{\Theta}$ for μ, $\mathbf{\Sigma_u}$, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{\Sigma_y}$ for simplicity.

Marginalizing $\mathbf{u}_i$ It is possible to analytically marginalize variables in this model: since the mean of $\mathbf{y}$ is linear in each $\mathbf{u}_i$ and all of these variables are normally distributed, the joint distribution of $(\mathbf{y}, \mathbf{u}_1, \ldots, \mathbf{u}_L)$ is also multivariate normal. We will focus for most of the paper on marginalizing the random effects $\mathbf{u}_i$ for a single i in order to leverage the tree structure mentioned earlier, but return in Section 4 to the idea of marginalizing many effects. Locally, $\mathbf{u}_i$ and $\mathbf{y}$ form the conditional distribution $p(\mathbf{u}_i, \mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i}) = p(\mathbf{u}_i|\mathbf{\Theta})p(\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i}, \mathbf{u}_i)$. Marginalized MCMC rewrites this conditional distribution as $p(\mathbf{u}_i, \mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i}) = p(\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i})p(\mathbf{u}_i|\mathbf{\Theta}, \mathbf{y}, \mathbf{u}_{-i})$, which reverses the dependence between $\mathbf{u}_i$ and $\mathbf{y}$ [35]. During sampling, $\mathbf{u}_i$ is marginalized from the HMC procedure by using $p(\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i})$ as the likelihood function and $p(\mathbf{\Theta}, \mathbf{u}_{-i})$ as the distribution of latent variables. After HMC sampling, $\mathbf{u}_i$ is recovered through ancestral sampling from $p(\mathbf{u}_i|\mathbf{\Theta}, \mathbf{y}, \mathbf{u}_{-i})$ given posterior samples of $(\mathbf{\Theta}, \mathbf{u}_{-i})$. The reversal requires analytical forms of $p(\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i})$ and $p(\mathbf{u}_i|\mathbf{\Theta}, \mathbf{y}, \mathbf{u}_{-i})$, which can be obtained via standard marginalization and conditioning operations on multivariate normal distributions [e.g., 6]

$$\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i} \sim \mathcal{N}\left(\sum_{j\neq i} \mathbf{A}_j\mathbf{u}_j + \mathbf{A}_i\mu_i + \mathbf{b}, \mathbf{A}_i\mathbf{\Sigma}_{\mathbf{u}_i}\mathbf{A}_i^T + \mathbf{\Sigma_y}\right),$$
$$\mathbf{u}_i|\mathbf{\Theta}, \mathbf{y}, \mathbf{u}_{-i} \sim \mathcal{N}\left(\mu_i + \mathbf{M}\left(\mathbf{y} - \sum_{j\neq i} \mathbf{A}_j\mathbf{u}_j - \mathbf{A}_i\mu_i - \mathbf{b}\right), (\mathbf{I} - \mathbf{M}\mathbf{A}_i)\mathbf{\Sigma}_{\mathbf{u}_i}\right), \tag{2}$$

where $\mathbf{M} = \mathbf{\Sigma}_{\mathbf{u}_i}\mathbf{A}_i^T(\mathbf{A}_i\mathbf{\Sigma}_{\mathbf{u}_i}\mathbf{A}_i^T + \mathbf{\Sigma_y})^{-1}$. Marginalization introduces the benefit of sampling in a lower dimensional space, but the cost depends on the complexity of evaluating the log-density functions of these two distributions in order to run HMC.

Table 1: Time complexities of different HMC approaches for the submodel involved in marginalization. Initialization is done once before the HMC loop. The log density is computed within each step of the leapfrog integrator. Recovery is performed for each sample from HMC. N is the number of observations, M is the dimension for one class of random effects, D is the dimension for all classes of random effects, L is the number of classes, d is the dimension for an effect of a group in a class.

Submodel	Approach	Initialization	Log density	Recovery
$p(\mathbf{u}_i, \mathbf{y}\vert\mathbf{\Theta}, \mathbf{u}_{-i})$	No marginalization	-	$O(Md^2 + NLd)$	-
	Naive marginalization	-	$O(M^3 + N^3)$	$O(M^3 + N^3)$
	Marginalize with lemmas	-	$O(Md^2 + NLd + Nd^2)$	$O(Md^2 + NLd + Nd^2)$
$p(\mathbf{v}, \mathbf{y}\vert\mathbf{\Theta})$	No marginalization	-	$O(Dd^2 + NLd)$	-
	Naive marginalization	-	$O(D^3 + N^3)$	$O(D^3 + N^3)$
	Marginalize with assumptions	$O(D^3 + NL^2d^2)$	$O(D^2 + NLd)$	$O(D^2 + NLd)$

2.1 Challenges of multivariate marginalization

In practice, the original model usually has structure that makes evaluating its density very efficient, which is lost by naive marginalization. For example, the observations in $\mathbf{y}$ are usually conditionally independent, making $\mathbf{\Sigma_y}$ diagonal; also, $\mathbf{\Sigma_{u}}_i$ is usually block diagonal with blocks of size $d \times d$. So evaluating the density $p(\mathbf{u}_i, \mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i}) = p(\mathbf{u}_i|\mathbf{\Theta})p(\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{1:L})$ requires $O(k_i d^3 + NLd) = O(M_i d^2 + NLd)$ time with the main operations being (1) inverting and computing the determinant of $\mathbf{\Sigma_u}$ and $\mathbf{\Sigma_y}$; (2) computing the mean parameter of $\mathbf{y}$. When $\mathbf{\Sigma_{u}}_i$ is diagonal, the complexity goes down to $O(M_i d + NLd)$. However, it is more expensive to evaluate the density of the reversed model in Equation (2). Computing $p(\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i})$ and $p(\mathbf{u}_i|\mathbf{\Theta}, \mathbf{y}, \mathbf{u}_{-i})$ requires the inverting and computing the determinant of the $N \times N$ matrix $\mathbf{A}_i\mathbf{\Sigma_{u}}_i\mathbf{A}_i^T + \mathbf{\Sigma_y}$, which we denote by $\mathbf{E}$ for simplicity. For the log likelihood, we need to compute $\log p(\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i}) = -\frac{1}{2}\det(\mathbf{E}) - \frac{1}{2}\mathbf{z}^T\mathbf{E}^{-1}\mathbf{z} + C$, where $\mathbf{z} = \mathbf{y} - \sum_{j \neq i}\mathbf{A}_j\mathbf{u}_j - \mathbf{A}_i\boldsymbol{\mu}_i - \mathbf{b}$. $\mathbf{E}$ is not diagonal and without using additional structure will trigger $O(N^3)$ operations within each step of the leapfrog integrator within HMC. For the recovery distribution $p(\mathbf{u}_i|\mathbf{\Theta}, \mathbf{y}, \mathbf{u}_{-i})$, $\mathbf{E}$ will be inverted when calculating $\mathbf{M}$. Also, a Cholesky decomposition for the covariance $(\mathbf{I} - \mathbf{MA}_i)\mathbf{\Sigma_{u}}_i$ should be computed for sampling, which takes $O(M_i^3)$ time. These cubic time operations are prohibitively expensive for large datasets. We summarize the complexities of different approaches in Table 1. In Section 3, we discuss how to marginalize one group of random effects with lemmas from linear algebra. In Section 4, we discuss how to marginalize all random effects with additional assumptions.

3 Marginalization with fast linear algebra

We now show how to speed up calculations with the marginalized model using fast linear algebra methods. In particular, we use the matrix inversion lemma and matrix determinant lemma together with special structure in the relevant matrices. In this section, we sometimes omit the subscript i such as for $\mathbf{A}_i$ and $\mathbf{\Sigma_{u}}_i$ for simplicity. The steps in log density evaluation and recovery are summarized in Algorithm 1, and in Algorithm 2 in the appendix, with comments about their implementation and cost. We mainly use sparsity and tree-structure in $\mathbf{A}$ to make operations faster. As an overview, computing $\mathbf{z}$ takes $O(NLd)$ time for L sparse matrix multiplications of time $O(Nd)$ each. Also, evaluating $\mathbf{As}$ and $\mathbf{A}^T\mathbf{t}$ both take $O(Nd)$ for any $\mathbf{s} \in \mathbb{R}^M$ and any $\mathbf{t} \in \mathbb{R}^N$. With tree-structure, we will see that $\mathbf{A}^T\mathbf{\Sigma}_\mathbf{y}^{-1}\mathbf{A}$ is block-diagonal and can be computed efficiently.

3.1 Matrix inversion and determinant lemmas in marginalization

The two main bottlenecks when evaluating $\log p(\mathbf{y}|\mathbf{\Theta}, \mathbf{u}_{-i})$ are computing $\det(\mathbf{E})$ and $\mathbf{z}^T\mathbf{E}^{-1}\mathbf{z}$. With the matrix determinant lemma [32], we have that

$$\det(\mathbf{E}) = \det(\mathbf{A\Sigma_u A}^T + \mathbf{\Sigma_y}) = \det(\mathbf{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T\mathbf{\Sigma}_\mathbf{y}^{-1}\mathbf{A})\det(\mathbf{\Sigma_u})\det(\mathbf{\Sigma_y}). \quad (3)$$

By the matrix inversion lemma or the Woodbury formula [53] we have that

$$\mathbf{E}^{-1} = (\mathbf{A\Sigma_u A}^T + \mathbf{\Sigma_y})^{-1} = \mathbf{\Sigma}_\mathbf{y}^{-1} - \mathbf{\Sigma}_\mathbf{y}^{-1}\mathbf{A}(\mathbf{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T\mathbf{\Sigma}_\mathbf{y}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{\Sigma}_\mathbf{y}^{-1}.$$

Therefore,

$$\mathbf{z}^T\mathbf{E}^{-1}\mathbf{z} = \mathbf{z}^T\mathbf{\Sigma}_\mathbf{y}^{-1}\mathbf{z} - \mathbf{z}^T\mathbf{\Sigma}_\mathbf{y}^{-1}\mathbf{A}(\mathbf{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T\mathbf{\Sigma}_\mathbf{y}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{\Sigma}_\mathbf{y}^{-1}\mathbf{z}. \quad (4)$$

Algorithm 1 Evaluating $\log p(\mathbf{y}|\boldsymbol{\Theta}, \mathbf{u}_{-i})$. Each $\mathbf{A}_i$ is an $N \times M_i$ sparse matrix with Nd elements and tree structure. $\boldsymbol{\Sigma}_\mathbf{y}$ is $N \times N$ diagonal. $\boldsymbol{\Sigma}_\mathbf{u}$ is $M \times M$ block-diagonal with block size d.

```
1: $\mathbf{z} = \mathbf{y} - \sum_{j \neq i} \mathbf{A}_j \mathbf{u}_j - \mathbf{A}_i \boldsymbol{\mu}_i - \mathbf{b}$    ▷Sparse matrix multiplication in $O(NLd)$ time
2: $\mathbf{F} = \boldsymbol{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$    ▷Block diagonal computation in $O((M + N)d^2)$ time
3: $\mathbf{x} = \mathbf{A}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{z}$    ▷Sparse matrix multiplication in $O(Nd)$ time
4: $a = \log\det(\mathbf{F}) + \log\det(\boldsymbol{\Sigma}_\mathbf{u}) + \log\det(\boldsymbol{\Sigma}_\mathbf{y})$    ▷Determinants in $O(Md^2)$ time
5: $b = \mathbf{z}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{z} - \mathbf{x}^T \mathbf{F}^{-1} \mathbf{x}$    ▷Quadratic form in $O(N + Md)$ time
6: return $-\frac{1}{2}(a + b) + C$
```

By using the facts that $\boldsymbol{\Sigma}_\mathbf{u}$ is block-diagonal, $\boldsymbol{\Sigma}_\mathbf{y}$ is diagonal, and $\mathbf{A}$ has Nd nonzero elements, the quantities $\det(\boldsymbol{\Sigma}_\mathbf{u})$, $\det(\boldsymbol{\Sigma}_\mathbf{y})$, $\mathbf{z}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{z}$, and $\mathbf{A}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{z}$ can each be calculated in $O(Md^2 + Nd)$ time. Equations (3) and (4) contain the expressions $\mathbf{F}^{-1}$ or $\det(\mathbf{F})$ for the $M \times M$ matrix $\mathbf{F} := \boldsymbol{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$, which both require $O(M^3)$ time when done naively. The following theorem shows that these quantities can be computed in $O((M + N)d^2)$ for LMMs.

Theorem 1. *If $\boldsymbol{\Sigma}_\mathbf{y}$ is diagonal, $\boldsymbol{\Sigma}_\mathbf{u}$ is block-diagonal with blocks of size $d \times d$, then $\mathbf{F} = \boldsymbol{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$ is also block-diagonal with $d \times d$ blocks and computing $\mathbf{A}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$ takes $O(Nd^2)$.*

Proof. The proof uses the tree-structure in $\mathbf{A}$. For details, see Appendix B.1. □

Therefore, it is $O((M + N)d^2)$ to compute $\det(\mathbf{F})$ and $\mathbf{F}^{-1}$. Combined with other parts in the formulas, the overall complexity is $O(Md^2 + NLd + Nd^2)$. In LMMs, d is usually small, so the complexity with marginalization can be viewed as the same as the complexity without marginalization.

3.2 Speeding up the recovery step

Different from evaluating $\log p(\mathbf{y}|\boldsymbol{\Theta}, \mathbf{u}_{-i})$, ancestral sampling from $p(\mathbf{u}_i|\boldsymbol{\Theta}, \mathbf{y}, \mathbf{u}_{-i})$ is only performed once for each posterior sample. When sampling from $p(\mathbf{u}_i|\boldsymbol{\Theta}, \mathbf{y}, \mathbf{u}_{-i})$, computing $\mathbf{M}$ directly is also costly. With the matrix inversion lemma, we have

$$\begin{aligned}\mathbf{M} &= \boldsymbol{\Sigma}_\mathbf{u}\mathbf{A}^T(\mathbf{A}\boldsymbol{\Sigma}_\mathbf{u}\mathbf{A}^T + \boldsymbol{\Sigma}_\mathbf{y})^{-1} \\ &= \boldsymbol{\Sigma}_\mathbf{u}\mathbf{A}^T\boldsymbol{\Sigma}_\mathbf{y}^{-1} - \boldsymbol{\Sigma}_\mathbf{u}\mathbf{A}^T\boldsymbol{\Sigma}_\mathbf{y}^{-1}\mathbf{A}(\boldsymbol{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T\boldsymbol{\Sigma}_\mathbf{y}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\boldsymbol{\Sigma}_\mathbf{y}^{-1}.\end{aligned} \tag{5}$$

With this expression, the mean variable $\boldsymbol{\mu} + \mathbf{Mz}$, then is evaluated in $O((M + N)d^2)$, by computing $\boldsymbol{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$ in the same way as Line 2 of Algorithm 1. For the covariance variable $(\mathbf{I} - \mathbf{MA})\boldsymbol{\Sigma}_\mathbf{u}$, we have from the reversed application of the matrix inversion lemma that

$$\begin{aligned}(\mathbf{I} - \mathbf{MA})\boldsymbol{\Sigma}_\mathbf{u} &= \boldsymbol{\Sigma}_\mathbf{u} - \boldsymbol{\Sigma}_\mathbf{u}\mathbf{A}^T(\mathbf{A}\boldsymbol{\Sigma}_\mathbf{u}\mathbf{A}^T + \boldsymbol{\Sigma}_\mathbf{y})^{-1}\mathbf{A}\boldsymbol{\Sigma}_\mathbf{u} \\ &= (\boldsymbol{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T\boldsymbol{\Sigma}_\mathbf{y}^{-1}\mathbf{A})^{-1}.\end{aligned}$$

Note that $\mathbf{F} = \boldsymbol{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$ is all block diagonal. For a block diagonal matrix with k blocks of size $d \times d$, the time complexity for a Cholesky decomposition is $O(kd^3) = O(Md^2)$. Combined with the complexity of computing $\mathbf{z}$, the recovery step takes $O(Md^2 + NLd + Nd^2)$ time.

4 Marginalizing multiple effects with additional assumptions

We have shown that it is efficient to marginalize one class of random effects. With additional practical assumptions, it is possible to marginalize all classes of random effects for efficient HMC inference. Instead of separating different classes of random effects, LMMs can also be written as $\mathbf{v} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_\mathbf{v})$, $\mathbf{y} \sim \mathcal{N}(\mathbf{Bv} + \mathbf{b}, \boldsymbol{\Sigma}_\mathbf{y})$, where $\mathbf{B} = [\mathbf{A}_1, ..., \mathbf{A}_L]$ and $\mathbf{v} = [\mathbf{u}_1^T, ..., \mathbf{u}_L^T]^T$. We define that $D = \sum_{i=1}^L M_i$. The matrix inversion and determinant lemmas can still be applied to marginalize $\mathbf{v}$ out, but the combined matrix $\mathbf{B}$ does not have the special structure of $\mathbf{A}_i$ we exploited in Section 3. More specifically, the computation of $\det(\mathbf{F})$ and the evaluation of $\mathbf{F}^{-1}$ for $\mathbf{F} = \boldsymbol{\Sigma}_\mathbf{v}^{-1} + \mathbf{B}^T \boldsymbol{\Sigma}_\mathbf{y}^{-1} \mathbf{B}$ both become non-trivial. We introduce additional assumptions to show that they can be solved faster in some special cases. For the general case, see the discussion section.

The assumption we make is that $\mathbf{\Sigma_v} = \tau_{\mathbf{v}}\mathbf{I}$ and $\mathbf{\Sigma_y} = \tau_{\mathbf{y}}\mathbf{I}$, where $\tau_{\mathbf{v}}, \tau_{\mathbf{y}}$ are scalars that either belong to $\mathbf{\Theta}$ or are fixed non-random parameters. This means that all effects share the same variance and all observations share the same noise scale. These assumptions are not as restrictive as it may appear. If the underlying distribution is $\mathbf{u}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2\mathbf{I})$ where σ_i is a fixed parameter, it is possible to reparameterize this distribution as $\mathbf{u}_i' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{A}_i' = \sigma_i\mathbf{A}_i$, $\mathbf{b}' = \mathbf{b} + \mathbf{B}\boldsymbol{\mu}_i$, and use $\mathbf{u}_i', \mathbf{A}_i', \mathbf{b}'$ in place of $\mathbf{u}_i, \mathbf{A}_i, \mathbf{b}$. Then $\mathbf{\Sigma_v}$ becomes a scaled identity matrix. Also, in many models, the noise scale for different observations is the same, making $\mathbf{\Sigma_y}$ a scaled identity matrix as well.

In practice, if the assumptions are satisfied, marginalization can be done in $O(D^2 + Nd)$ time with $O(D^3 + NL^2d^2)$ preprocessing. Details are provided in Appendix B.3.

5 Related Work

While many works aim to improve HMC directly [71, 30, 58, 73], a number of other works focus on model transformation. Non-centered parameterization [49] is a widely used trick among MCMC users to alleviate slow sampling in difficult posterior distributions. However, there is no general way to know whether a non-centered parameterization will be beneficial [76]. Variationally inferred parameterization [26] proposes to learn a model parameterization from a specified family that will lead to effective sampling. In Parno and Marzouk [52] and Hoffman et al. [33], preconditioners for HMC are learned to transform the model to be approximately isotropic Gaussians. Marginalization differs from reparameterization in that it reduces the problem dimension as well as potentially alleviating difficult characteristics such as funnels, so it has two mechanisms to improve MCMC efficiency. The Laplace approximation (LA) is one way to approximately marginalize variables in MCMC [59, 40, 65], but it may be difficult to quantify the error or recover the marginalized variables.

Marginalization, or Rao-Blackwellization, has been an important topic in Bayesian inference and probabilistic programming. In Gibbs sampling, marginalization is usually called collapsing [37]. Collapsed Gibbs sampling has been developed for latent Dirichlet allocation [56] and LMMs [50]. We explore marginalization in the context of HMC, which induces different considerations. Methods with HMC do not have to make the conditional distributions of the marginalized model tractable. Marginalization is also related to symbolic inference in probabilistic programming. Hakaru [44] and PSI [21, 22] are systems for performing exact Bayesian inference by symbolically marginalizing all latent variables. To marginalize discrete variables, Gorinova et al.[27] propose an information flow type system. Another line of related work is delayed sampling [43, 3], which automates marginalization of variables within Rao-Blackwellized particle filters [42]. Lai et al. [35] developed an automatic system for marginalizing variables in HMC, but is limited to scalar variables so cannot leverage vectorization and forces users to write models with univariate distributions.

Linear algebra tricks have been widely utilized in various machine learning algorithms, such as ridge regression [68], Gaussian processes [61] and Kalman filters [60]. Recently, frameworks [62, 20, 57] have been proposed to ease the implementation of fast linear algebras in machine learning algorithms. Marginalization in Bayesian models may be an interesting application of those frameworks.

Fast and scalable inference for LMMs has been studied in the context of maximum likelihood estimation [19], variational EM [24], Gibbs sampling [51] and numerical integration [28]. We are the first to consider speeding up the inference of LMMs with HMC. There is also a recent trend in integrating random effects into deep neural networks for correlated data [67] or personalization [66, 64, 74] with parameters estimated by maximum likelihood.

6 Experiments

We conduct experiments on LMMs from various disciplines using the default no-U-turn sampler (NUTS) [34] from NumPyro [5, 54], which has an adaptive step size with dual averaging, adaptive and diagonal mass matrix, target acceptance probability of 0.8, and maximum tree depth of 10. For the ETH instructor evaluation model, we set the maximum tree depth to 12 to overcome difficulties performing inference without marginalization in preliminary experiments. For all models, we use weakly informative priors unless specified. In general, our conclusion is insensitive to the choice of hyperparameters and priors. For all experiments, we collect 10,000 warm up samples for tuning, and 100,000 samples for evaluation, and evaluate performance via effective sample size (ESS) and running time.

Table 2: Running time in seconds for HMC, with or without marginalization. Mean and standard deviation over 5 independent runs are reported. Experiments are run on NVIDIA A40.

Method	No marginalization	Marginalize $\mathbf{u}_1$	Marginalize $\mathbf{u}_2$	Marginalize $\mathbf{u}_3$	Marginalize $\mathbf{u}$
Time (s)	13417 (98)	5004 (1468)	2607 (3)	3071 (4)	**631** (12)

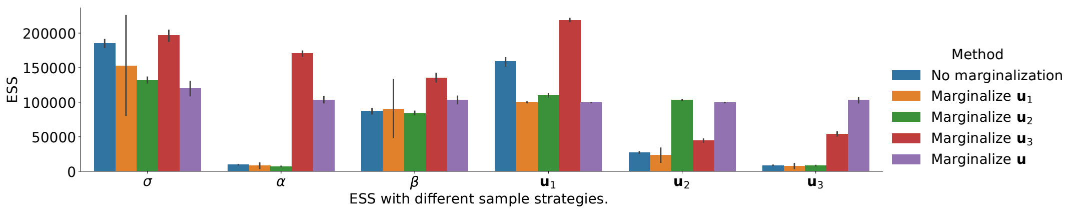

Figure 2: Average ESS for each variable on the instruction evaluation model with different HMC strategies. Numbers above the sample size 100,000 indicate effective sampling.

6.1 Marginalization in cross-effects models

Cross-effects models are a type of LMM that have more than one class of random effects (i.e. $L > 1$). Usually each observation belongs to one subject group (e.g. individuals, animals) and one item group (e.g. questions, objects). The correlation among latent variables can create severely challenging geometry that slows down the sampling of HMC. With our idea, it is possible to marginalize one or more group of effects from the model, reducing the dimension of latent space for faster sampling and better geometry.

ETH instructor evaluations An example cross-effects model describes university lecture evaluations by students at ETH [4]. The dataset records $N = 73421$ ratings, where each rating y_n comes from student s_n for professor p_n teaching a course from department d_n, with t_n indicating whether the professor is teaching outside their own department. There are a total of $M_1 = 2972$ students, $M_2 = 1128$ professors and $M_3 = 14$ departments. We use a version of the model from the document of Tensorflow probability [13]. The model is

$$\text{Likelihood} : y_n \sim N(u_{1,s_n} + u_{2,p_n} + u_{3,d_n} + \alpha + \beta t_n, \sigma^2),$$

$$\text{Prior} : u_{1,i} \sim N(0, 1),\ u_{2,j} \sim N(0, 1),\ u_{3,k} \sim N(0, 1),\ \alpha \sim N(0, 5),\ \beta \sim N(0, 1),\ \sigma \sim N^+(0, 1),$$

where $1 \le i \le M_1$, $1 \le j \le M_2$ and $1 \le k \le M_3$. Given the dataset, we wish to learn about the latent variables $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$, α, β and σ. HMC is the most direct way to sample those variables, but the dimension and complicated relations make it inefficient. Marginalization can be applied to one of the effects, $\mathbf{u}_1$, $\mathbf{u}_2$ or $\mathbf{u}_3$. We report the running time of sampling from the model with and without marginalization in Table 2. We found that marginalizing any group of random effects improves the sampling speed of HMC. However, the improvements are not necessarily predicted by the dimension of marginalized variable: HMC is faster when marginalizing $\mathbf{u}_3$ than when marginalizing $\mathbf{u}_1$ even though $\mathbf{u}_1$ has 200-times higher dimension than $\mathbf{u}_3$. In Figure 2, the ESS for each variable is reported. Without marginalization, sampling $\mathbf{u}_2$ and $\mathbf{u}_3$ are both difficult compared to sampling $\mathbf{u}_1$, and HMC becomes more efficient when marginalizing either of these variables, so we conjecture that $\mathbf{u}_2$ and $\mathbf{u}_3$ are responsible for the difficulty for sampling in the original model. In this model, all random effects are independent and have the same variance, so $\mathbf{\Sigma_u}$ is a scaled identity matrix and we can marginalize all random effects efficiently. This approach is observed to be the most efficient in our experiments, despite having quadratic complexity in D. Overall, marginalization never hurts ESS, and runs faster. We expect that any marginalization strategy works better than HMC in the original model, a finding which will be consistent across experiments. Additional results of this experiment, including trace plots and $\hat{R}$ diagnosis, are included in Figure 6 and Table 5 in the Appendix.

6.2 Marginalization vs reparameterization

To tackle bad geometry in statistical models, another model transformation is non-centered parameterization, or reparameterization [49]. Reparameterization converts the distribution of $z \sim N(\mu, \sigma^2)$ into $\epsilon \sim N(0, 1)$ and $z = \epsilon\sigma + \mu$. Reparameterization is especially useful for funnel shapes in hierarchical models. We note that when applicable, marginalization is able to solve a broader class of problems. We compare marginalization and reparameteriation on the grouse ticks model.

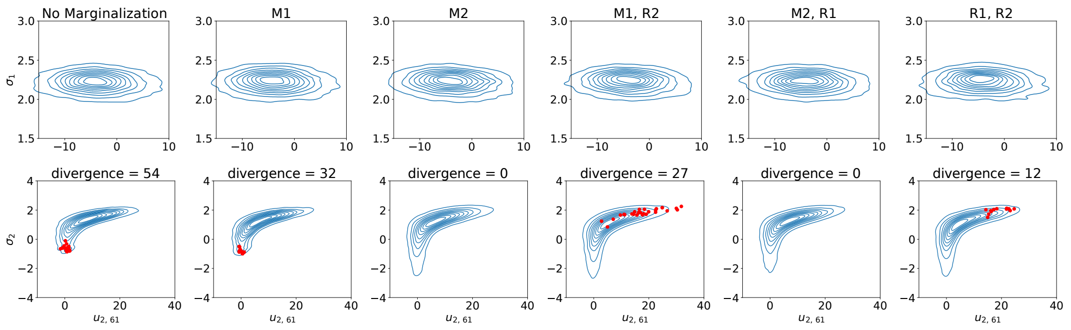

Figure 3: Distribution of 10,000 samples for variable pairs $(\sigma_1, u_{1,1})$ and $(\sigma_2, u_{2,61})$ on the grouseticks model with different methods. We use M1 to represent marginalizing $\mathbf{u}_1$, M2 to represent marginalizing $\mathbf{u}_2$, R1 to represent reparameterizing $\mathbf{u}_1$, R2 to represent reparameterizing $\mathbf{u}_2$. The number of divergences for each case are reported, with locations shown as red dots. We choose $u_{2,61}$ to demonstrate the distribution of divergences when reparameterizing $\mathbf{u}_2$.

Table 3: Compilation time T_c and running time T_r in seconds for marginalized MCMC [35], with or without vectorization. Mean and std across 5 independ runs are reported.

Model	T_c of [35]	T_r of [35]	T_c of ours	T_r of ours
Electric company	552 (4)	1249 (95)	7 (0)	252 (23)
Pulmonary fibrosis	727 (11)	2208 (80)	10 (1)	178 (3)

Grouse ticks The dataset [4] contains observations $\mathbf{y}$ of the the number of ticks on the heads of red grouse chicks in the field. Each observation y_k comes from brood b_k in location l_k during year e_k at altitude a_k, where year and altitude give fixed effects, and there are random effects $\mathbf{u}_1$ and $\mathbf{u}_2$ corresponding to brood and location. There are $N = 403$ observations, $M_1 = 118$ broods and $M_2 = 63$ locations. We define the hierarchical model as follows:

$$\text{Likelihood}: y_k \sim \mathcal{N}(u_{1,b_k} + u_{2,l_k} + \beta_e e_k + \beta_a a_k, \sigma_t^2)$$
$$\text{Prior}: \mu_1 \sim \mathcal{N}(0,1),\ \sigma_1 \sim \text{HalfCauchy}(5),\ \mu_2 \sim \mathcal{N}(0,1),\ \sigma_2 \sim \text{HalfCauchy}(5),$$
$$\beta_e \sim \mathcal{N}(0,1),\ \beta_a \sim \mathcal{N}(0,1),\ u_{1,i} \sim \mathcal{N}(\mu_1, \sigma_1^2),\ u_{2,j} \sim \mathcal{N}(\mu_2, \sigma_2^2),\ \sigma_t \sim \text{HalfCauchy}(5),$$

where $i = 1, \ldots, M_1, j = 1, \ldots, M_2, k = 1, \ldots, N$ and each y_k is observed. The correlation between σ and $\mathbf{u}$ creates the funnel shape that makes vanilla HMC inefficient. Nevertheless, it is possible to apply either marginalization or reparameterization to each random effect. In Figure 3, we plot the distributions of samples for variable pairs $(\sigma_1, u_{1,1})$ and $(\sigma_2, u_{2,1})$ with different combinations of marginalization and reparameterization. There is a difficult correlation between σ_2 and $\mathbf{u}_2$. After applying marginalization or reparameterization to $\mathbf{u}_2$, HMC manages to explore the funnel region (at low values of σ_1). However, we find that only samplers that marginalize $\mathbf{u}_2$ report zero divergent transitions after warm-up. Such behavior is consistent with different random seeds. See Table 6 in the Appendix. Also, the distribution of divergent samples is related to specific parameters when reparameterizing $\mathbf{u}_2$, implying that reparameterization introduces pathologies that create challenges for HMC inference. In addition, we find that reparameterization does not improve the running time of HMC, while marginalizing $\mathbf{u}_2$ speeds up sampling by about 20%.

6.3 Benefits from vectorization

In theory, marginalization with LMMs can be done by constructing a graphical model for scalar random variables and performing automatic marginalization as in [35]. But it is more efficient to marginalize in a vectorized way. We demonstrate the benefits from vectorization in Table 3.Both marginalization strategies are performed on two hierarchical linear regression models, the electric company model [23] and the pulmonary fibrosis model [63]. We find that vectorized marginalization is much more efficient for sampling from the two models.

Table 4: Specifications of the datasets from cognitive sciences. Details of each model are provided in Appendix D. GPU models run on an NVIDIA RTX 2080ti GPU. CPU models run on one Intel Xeon Gold 6148 processor.

	dillonE1[12]	dutch[17]	eeg[48]	english[69]	gg05[29]	mandarin[75]	mandarin2[70]	pupil[72]	stroop[16]
N	2855	372	26176	768	672	547	595	2228	3058
L	2	2	2	2	2	2	2	1	1
M_1	40	24	334	48	42	37	40	20	50
M_2	48	16	80	16	16	15	15	-	-
Likelihood	LogNormal	Normal	Normal	Normal	LogNormal	LogNormal	LogNormal	Normal	LogNormal
Device	GPU	GPU	GPU	CPU	CPU	CPU	GPU	GPU	GPU

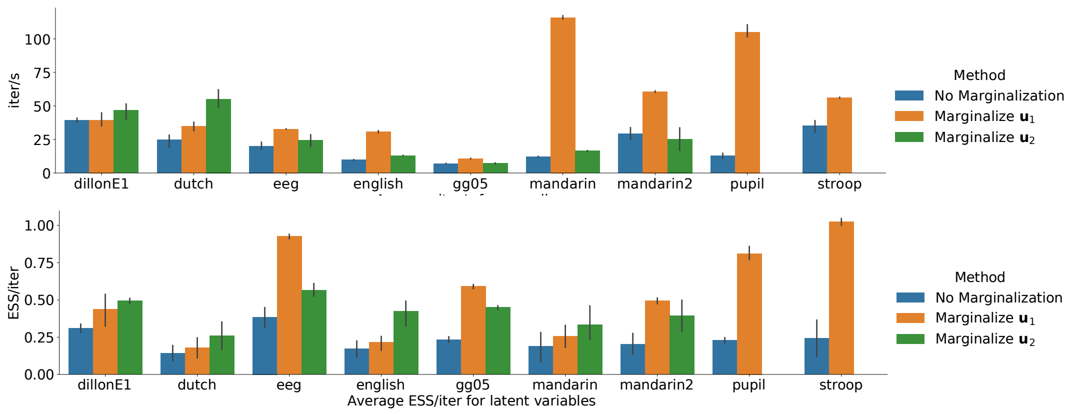

Figure 4: Experimental results for the 9 cognitive science datasets with and without marginalization. Each experiment is performed 5 times with different random seeds. Marginalization usually improves sampling speed measured by iterations per second (iter/s) and sample efficiency measured by ESS per iteration (ESS/iter).

6.4 Applications in cognitive sciences

Hierarchical Bayesian inference with LMMs has wide applications in cognitive science [47]. We highlight the effectiveness of marginalization with 9 datasets from cognitive science (Table 4). They cover various settings, with one or two random effects, normal or log-normal likelihoods, on CPU or GPU. Experiments that are slow on CPU are performed on GPU. Each dataset corresponds to an LMM where both the intercept and the coefficient include random effects. Details of all the models can be found in Appendix D. Results are summarized in Figure 4. Marginalization usually improves the sampling speed of HMC and consistently improves efficiency measured by ESS per iteration.

7 Discussion

There are several promising directions for future work.

7.1 Marginalization vs Rao-Blackwellization

Marginalization is related to Rao-Blackwellization. This paper focuses on marginalization, which improves the speed of obtaining samples from the remaining variables by improving mixing times, reducing the cost per iteration, or both. Combining marginalization with Rao-Blackwellization is an interesting avenue for future work. More formally, if one is interested in some expectation $\mathbb{E}_{(\boldsymbol{\Theta},\mathbf{u})\sim p(\boldsymbol{\Theta},\mathbf{u}|\mathbf{y})}[f(\boldsymbol{\Theta},\mathbf{u})]$ in an LMM, there is a Monte Carlo estimator

$$E_1 = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{\Theta}^i, \mathbf{u}^i),$$

where $(\boldsymbol{\Theta}^i, \mathbf{u}^i) \sim p(\boldsymbol{\Theta}, \mathbf{u}|\mathbf{y})$ and N is the sample size. Marginalization is a trick to improve the efficiency of the posterior sampling, so that we can achieve the same estimation variance with smaller N or less runtime . At the same time, we also have access to a conditional distribution that is useful for Rao-Blackwellization. If the effects variable $\mathbf{u}$ can be marginalized we have both an approximate posterior for $p(\boldsymbol{\Theta}|\mathbf{y})$ and an analytical conditional distribution $p(\mathbf{u}|\boldsymbol{\Theta}, \mathbf{y})$. With Rao-Blackwellization we have that $\mathbb{E}_{(\boldsymbol{\Theta},\mathbf{u})\sim p(\boldsymbol{\Theta},\mathbf{u}|\mathbf{y})}[f(\boldsymbol{\Theta}, \mathbf{u})] = \mathbb{E}_{\boldsymbol{\Theta}\sim p(\boldsymbol{\Theta}|\mathbf{y})}[\mathbb{E}_{\mathbf{u}\sim p(\mathbf{u}|\boldsymbol{\Theta},\mathbf{y})}[f(\boldsymbol{\Theta}, \mathbf{u})]]$. In such case, another Monte Carlo estimator can be constructed:

$$E_2 = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\mathbf{u}\sim p(\mathbf{u}|\boldsymbol{\Theta},\mathbf{y})} f(\boldsymbol{\Theta}^i, \mathbf{u}) ,$$

where $\boldsymbol{\Theta}^i \sim p(\boldsymbol{\Theta}|\mathbf{y})$. For some functions, such as those that are polynomial in $\mathbf{u}$, the inner expectation can be computed exactly using properties of Gaussians. In other cases, the inner expectation can be estimated cheaply via Monte Carlo using exact samples from $p(\mathbf{u}|\boldsymbol{\Theta}_i, \mathbf{y})$.

7.2 Marginalizing multiple effects in general models

In Section 4, we proposed to marginalize multiple classes of random effects by assuming a scaled identity covariance matrix. To marginalize multiple effects in general models, a possibility is to compute $\mathbf{z}^T \mathbf{E}^{-1} \mathbf{z}$ and estimate $\det(\mathbf{E})$ and the corresponding gradients with conjugate gradient (CG) solvers [14, 20]. However, this approach uses stochastic estimators for the determinant and gradients, which introduce bias into the HMC dynamics. These biases can be corrected through pseudo-marginalization [2], but it is unclear how significantly the extra stochasticity will affect the sampling. Another possible way to marginalize multiple effects for LMMs is to introduced the balanced levels assumption [50]. We leave these ideas for future exploration.

7.3 Beyond normal likelihoods

In this work, we only consider normal or log-normal likelihoods, but our method can be easily generalized to other deterministic transformation of normal likelihood. This implies that marginalization can benefit regression with most continuous predictors given proper link functions. Another potential future direction is to marginalize classification models with probit regressions [1]. Marginalization will turn probit models into multivariate probit models as $\mathbf{A}\boldsymbol{\Sigma}_{\mathbf{u}}\mathbf{A}^T + \boldsymbol{\Sigma}_{\mathbf{y}}$ is a dense covariance matrix, which may require a simulation-based method [10] or variational Bayes [39]. It will be interesting to see how ideas from multivariate probit regression could be fit into an HMC pipeline. In a broader context, marginalization is related to data augmentation techniques that "create" conjugacy for non-normal likelihoods or non-normal effects. Those techniques were developed for Gibbs sampling, e.g. [18, 55], but may also be useful for HMC.

7.4 Integration with probabilistic programming

We have developed a tool to speed up the HMC inference for LMMs. In our implementation, the marginalized likelihood $p(\mathbf{y}|\boldsymbol{\Theta}, \mathbf{u}_{-i})$ is defined as a special type of parametric distribution available to the user, and the recovery distribution $p(\mathbf{u}_i|\boldsymbol{\Theta}, \mathbf{u}_{-i}, \mathbf{y})$ is a function called after sampling. In our experiments, marginalization never hurt sampling efficiency measured by ESS/s, and usually helped. Thus, it would be desirable to always marginalize one group of random effects when the model is an LMM. Future work could aim to automatically apply such transformations to user-specified LMMs. There are two possible high-level approaches. The first is to perform marginalization starting with a model described using a high-level abstraction such as an R formula. Then, when compiling the high-level model description into a concrete model (e.g., a probabilistic program), we can marginalize one or more of the effects using our methods. The second is to perform marginalization starting with a user-written probabilistic program representing an LMM. In this case, some compilation or program tracing technique will be needed to convert the user's program to a model representation suitable for manipulation. For example, Lai et al. [35] used program tracing to construct a graphical model representation that could be programmatically analyzed and transformed. To apply this methodology to LMMs, a special parser would also be needed to match the models to LMMs.

Acknowledgement

The authors thank Yuling Yao and the anonymous reviewers for comments that greatly improved the manuscript. This material is based upon work supported by the National Science Foundation under Grants #1749854, #2045900.

References

[1] Alan Agresti. *Foundations of linear and generalized linear models*. John Wiley & Sons, 2015.

[2] Christophe Andrieu and Gareth O Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. 2009.

[3] Eric Atkinson, Charles Yuan, Guillaume Baudart, Louis Mandel, and Michael Carbin. Semi-symbolic inference for efficient streaming probabilistic programming. *Proceedings of the ACM on Programming Languages*, 6(OOPSLA2):1668–1696, 2022.

[4] Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. doi: 10.18637/jss.v067.i01.

[5] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *Journal of machine learning research*, 20(28):1–6, 2019.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] Helen Brown and Robin Prescott. *Applied mixed models in medicine*. John Wiley & Sons, 2014.

[8] Paul-Christian Bürkner. BRMS: An R package for Bayesian multilevel models using Stan. *Journal of statistical software*, 80:1–28, 2017.

[9] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76, 2017.

[10] Siddhartha Chib and Edward Greenberg. Analysis of multivariate probit models. *Biometrika*, 85(2):347–361, 1998.

[11] Michael Creutz. Global Monte Carlo algorithms for many-fermion systems. *Physical Review D*, 38(4):1228, 1988.

[12] Brian Dillon, Alan Mishler, Shayne Sloggett, and Colin Phillips. Contrasting intrusion profiles for agreement and anaphora: Experimental and modeling evidence. *Journal of Memory and Language*, 69(2):85–103, 2013.

[13] Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

[14] Justin Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pages 318–326. PMLR, 2012.

[15] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

[16] Charles R Ebersole, Olivia E Atherton, Aimee L Belanger, Hayley M Skulborstad, Jill M Allen, Jonathan B Banks, Erica Baranski, Michael J Bernstein, Diane BV Bonfiglio, Leanne Boucher, et al. Many labs 3: Evaluating participant pool quality across the academic semester via replication. *Journal of Experimental Social Psychology*, 67:68–82, 2016.

[17] Stefan L Frank, Thijs Trompenaars, and Shravan Vasishth. Cross-linguistic differences in processing double-embedded relative clauses: Working-memory constraints or language statistics? *Cognitive science*, 40(3):554–578, 2016.

[18] Sylvia Frühwirth-Schnatter, Rudolf Frühwirth, Leonhard Held, and Håvard Rue. Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data. *Statistics and Computing*, 19:479–492, 2009.

[19] Katelyn Gao and Art B Owen. Estimation and inference for very large linear mixed effects models. *Statistica Sinica*, 30(4):1741–1771, 2020.

[20] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in neural information processing systems*, 31, 2018.

[21] Timon Gehr, Sasa Misailovic, and Martin Vechev. PSI: Exact symbolic inference for probabilistic programs. In *Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I 28*, pages 62–83. Springer, 2016.

[22] Timon Gehr, Samuel Steffen, and Martin Vechev. λPSI: exact inference for higher-order probabilistic programs. In *Proceedings of the 41st acm sigplan conference on programming language design and implementation*, pages 883–897, 2020.

[23] Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press, 2006.

[24] Disha Ghandwani, Swarnadip Ghosh, Trevor Hastie, and Art B Owen. Scalable solution to crossed random effects model with random slopes. *arXiv preprint arXiv:2307.12378*, 2023.

[25] Edward Gibson and James Thomas. Memory limitations and structural forgetting: The perception of complex ungrammatical sentences as grammatical. *Language and Cognitive Processes*, 14(3):225–248, 1999.

[26] Maria Gorinova, Dave Moore, and Matthew Hoffman. Automatic reparameterisation of probabilistic programs. In *International Conference on Machine Learning*, pages 3648–3657. PMLR, 2020.

[27] Maria I Gorinova, Andrew D Gordon, Charles Sutton, and Matthijs Vákár. Conditional independence by typing. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 44(1):1–54, 2021.

[28] Philip Greengard, Jeremy Hoskins, Charles C Margossian, Jonah Gabry, Andrew Gelman, and Aki Vehtari. Fast methods for posterior inference of two-group normal-normal models. *Bayesian Analysis*, 18(3):889–907, 2023.

[29] Daniel Grodner and Edward Gibson. Consequences of the serial nature of linguistic input for sentenial complexity. *Cognitive science*, 29(2):261–290, 2005.

[30] Richard Grumitt, Biwei Dai, and Uros Seljak. Deterministic Langevin Monte Carlo with normalizing flows for Bayesian inference. *Advances in Neural Information Processing Systems*, 35:11629–11641, 2022.

[31] Xavier A Harrison, Lynda Donaldson, Maria Eugenia Correa-Cano, Julian Evans, David N Fisher, Cecily ED Goodwin, Beth S Robinson, David J Hodgson, and Richard Inger. A brief introduction to mixed effects modelling and multi-model inference in ecology. *PeerJ*, 6:e4794, 2018.

[32] David A Harville. Matrix algebra from a statistician's perspective, 1998.

[33] Matthew Hoffman, Pavel Sountsov, Joshua V Dillon, Ian Langmore, Dustin Tran, and Srinivas Vasudevan. Neutra-lizing bad geometry in Hamiltonian Monte Carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

[34] Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[35] Jinlin Lai, Javier Burroni, Hui Guan, and Daniel Sheldon. Automatically marginalized MCMC in probabilistic programming. In *International Conference on Machine Learning*, pages 18301–18318. PMLR, 2023.

[36] Daniel Lewandowski, Dorota Kurowicka, and Harry Joe. Generating random correlation matrices based on vines and extended onion method. *Journal of multivariate analysis*, 100(9): 1989–2001, 2009.

[37] Jun S Liu. The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966, 1994.

[38] Colin M MacLeod. Half a century of research on the Stroop effect: an integrative review. *Psychological bulletin*, 109(2):163, 1991.

[39] Stephan Mandt, Florian Wenzel, Shinichi Nakajima, John Cunningham, Christoph Lippert, and Marius Kloft. Sparse probit linear mixed model. *Machine Learning*, 106:1621–1642, 2017.

[40] Charles Margossian, Aki Vehtari, Daniel Simpson, and Raj Agrawal. Hamiltonian Monte Carlo using an adjoint-differentiated Laplace approximation: Bayesian inference for latent Gaussian models and beyond. *Advances in Neural Information Processing Systems*, 33:9086–9097, 2020.

[41] Lotte Meteyard and Robert AI Davies. Best practice guidance for linear mixed-effects models in psychological science. *Journal of Memory and Language*, 112:104092, 2020.

[42] Kevin Murphy and Stuart Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Sequential Monte Carlo methods in practice*, pages 499–515. Springer, 2001.

[43] Lawrence Murray, Daniel Lundén, Jan Kudlicka, David Broman, and Thomas Schön.Delayed sampling and automatic Rao-Blackwellization of probabilistic programs. In *International Conference on Artificial Intelligence and Statistics*, pages 1037–1046. PMLR, 2018.

[44] Praveen Narayanan, Jacques Carette, Wren Romano, Chung-chieh Shan, and Robert Zinkov. Probabilistic inference by program transformation in Hakaru (system description). In *Functional and Logic Programming: 13th International Symposium, FLOPS 2016, Kochi, Japan, March 4-6, 2016, Proceedings 13*, pages 62–79. Springer, 2016.

[45] Radford M Neal. Slice sampling. *The annals of statistics*, 31(3):705–767, 2003.

[46] Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2(11):2, 2011.

[47] Bruno Nicenboim, Daniel Schad, and Shravan Vasishth. An introduction to Bayesian data analysis for cognitive science. *Under contract with Chapman and Hall/CRC statistics in the social and behavioral sciences series*, 2021.

[48] Mante S Nieuwland, Stephen Politzer-Ahles, Evelien Heyselaar, Katrien Segaert, Emily Darley, Nina Kazanina, Sarah Von Grebmer Zu Wolfsthurn, Federica Bartolozzi, Vita Kogan, Aine Ito, et al. Large-scale replication study reveals a limit on probabilistic prediction in language comprehension. *ELife*, 7:e33468, 2018.

[49] Omiros Papaspiliopoulos, Gareth O Roberts, and Martin Sköld. A general framework for the parametrization of hierarchical models. *Statistical Science*, pages 59–73, 2007.

[50] Omiros Papaspiliopoulos, Gareth O Roberts, and Giacomo Zanella. Scalable inference for crossed random effects models. *Biometrika*, 107(1):25–40, 2020.

[51] Omiros Papaspiliopoulos, Timothée Stumpf-Fétizon, and Giacomo Zanella.Scalable Bayesian computation for crossed and nested hierarchical models. *Electronic Journal of Statistics*, 17(2): 3575–3612, 2023.

[52] Matthew D Parno and Youssef M Marzouk.Transport map accelerated Markov chain Monte Carlo. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):645–682, 2018.

[53] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

[54] Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv preprint arXiv:1912.11554*, 2019.

[55] Nicholas G Polson, James G Scott, and Jesse Windle. Bayesian inference for logistic models using Pólya–Gamma latent variables. *Journal of the American statistical Association*, 108(504): 1339–1349, 2013.

[56] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed Gibbs sampling for latent Dirichlet allocation.In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577, 2008.

[57] Andres Potapczynski, Marc Finzi, Geoff Pleiss, and Andrew G Wilson. CoLA: Exploiting compositional structure for automatic and efficient numerical linear algebra. *Advances in Neural Information Processing Systems*, 36, 2024.

[58] Jakob Robnik, G Bruno De Luca, Eva Silverstein, and Uroš Seljak. Microcanonical Hamiltonian Monte Carlo. *The Journal of Machine Learning Research*, 24(1):14696–14729, 2023.

[59] Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 71(2):319–392, 2009.

[60] Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*, volume 17.Cambridge university press, 2023.

[61] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

[62] Matthias Seeger, Asmus Hetzel, Zhenwen Dai, Eric Meissner, and Neil D Lawrence. Auto-differentiating linear algebra. *arXiv preprint arXiv:1710.08717*, 2017.

[63] Ahmed Shahin, Carmela Wegworth, David, Elizabeth Estes, Julia Elliott, Justin Zita, Simon-Walsh, Slepetys, and Will Cukierski. OSIC pulmonary fibrosis progression, 2020.

[64] Jun Shi, Chengming Jiang, Aman Gupta, Mingzhou Zhou, Yunbo Ouyang, Qiang Charles Xiao, Qingquan Song, Yi Wu, Haichao Wei, and Huiji Gao. Generalized deep mixed models. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3869–3877, 2022.

[65] Justin D Silverman, Kimberly Roche, Zachary C Holmes, Lawrence A David, and Sayan Mukherjee. Bayesian multinomial logistic normal models through marginally latent matrix-T processes. *Journal of Machine Learning Research*, 23(7):1–42, 2022.

[66] Giora Simchoni and Saharon Rosset. Using random effects to account for high-cardinality categorical features and repeated measures in deep neural networks. *Advances in Neural Information Processing Systems*, 34:25111–25122, 2021.

[67] Giora Simchoni and Saharon Rosset. Integrating random effects in deep neural networks. *Journal of Machine Learning Research*, 24(156):1–57, 2023.

[68] Wessel N van Wieringen. Lecture notes on ridge regression. *arXiv preprint arXiv:1509.09169*, 2015.

[69] Shravan Vasishth, Katja Suckow, Richard L Lewis, and Sabine Kern. Short-term forgetting in sentence comprehension: Crosslinguistic evidence from verb-final structures.*Language and Cognitive Processes*, 25(4):533–567, 2010.

[70] Shravan Vasishth, Zhong Chen, Qiang Li, and Gueilan Guo. Processing Chinese relative clauses: Evidence for the subject-relative advantage. *PloS one*, 8(10):e77006, 2013.

[71] Greg Ver Steeg and Aram Galstyan. Hamiltonian dynamics with non-Newtonian momentum for rapid sampling. *Advances in Neural Information Processing Systems*, 34:11012–11025, 2021.

[72] Basil Wahn, Daniel P Ferris, W David Hairston, and Peter König. Pupil sizes scale with attentional load and task experience in a multiple object tracking task. *PloS one*, 11(12): e0168087, 2016.

[73] Jun-Kun Wang and Andre Wibisono. Accelerating Hamiltonian Monte Carlo via Chebyshev integration time. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

[74] Torsten Wörtwein, Nicholas B Allen, Lisa B Sheeber, Randy P Auerbach, Jeffrey F Cohn, and Louis-Philippe Morency. Neural mixed effects for nonlinear personalized predictions. In *Proceedings of the 25th International Conference on Multimodal Interaction*, pages 445–454, 2023.

[75] HI Wu and E Gibson. Processing Chinese relative clauses in context. In *Poster presented at the 21st CUNY Conference on Sentence Processing, University of North Carolina at Chapel Hill*, 2008.

[76] Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Yes, but did it work?: Evaluating variational inference. In *International Conference on Machine Learning*, pages 5581–5590. PMLR, 2018.

[77] Zhaoxia Yu, Michele Guindani, Steven F Grieco, Lujia Chen, Todd C Holmes, and Xiangmin Xu. Beyond t test and ANOVA: applications of mixed-effects models for more rigorous statistical analysis in neuroscience research. *Neuron*, 110(1):21–35, 2022.

A Notation table

We summarize the important symbols used in the paper.

Symbols	Description
N	Number of observations, and dimension of $\mathbf{y}$
M, M_i	Dimension for all effects in one class of mixed effects
L	Number of classes of mixed effects
D	Dimension for all mixed effects
d	Dimension for effects of a group in a class
k, k_i	Number of groups in a class
α	Intercept for linear regression
β	Slope for linear regression
σ	Standard deviation
u, $\mathbf{u}$	Random effects
$\mathbf{v}$	Concatenated random effects
y, $\mathbf{y}$	Observations
c, t	Covariates, or treatments
$\mathbf{T}$	A prior variable sampled from half-normal distributions
$\mathbf{L}$	A prior variable sampled from LKJ distributions
g	Grouping variables
$\mathbf{\Theta}$	Global variables, including priors and fixed effects
μ	Mean of random effects
$\mathbf{A}$	Design matrix for random effects
$\mathbf{B}$	Concatenated design matrices
$\mathbf{b}$	Intercept term in the canonical form for LMMs
$\mathbf{\Sigma_u}$	Covariance matrix for a class of random effects
$\mathbf{\Sigma_y}$	Covariance matrix for the observations
$\mathbf{\Sigma_v}$	Covariance matrix for all random effects
$\tau_\mathbf{v}$	Scale for $\mathbf{\Sigma_v}$ with the scaled identity assumption
$\tau_\mathbf{y}$	Scale for $\mathbf{\Sigma_y}$ with the scaled identity assumption
$\mathbf{M}$	A shared matrix in the reversed model
$\mathbf{z}$	Difference between observation and mean of the marginalized likelihood
$\mathbf{E}$	A dense $N \times N$ matrix that is difficult to directly compute
$\mathbf{F}$	The core matrix after applying the two linear algebra lemmas
$\mathbf{G}$	An intermediate matrix in the implementation
$\mathbf{x}$	An intermediate vector in the implementation
$\mathbf{r}$	A row of $\mathbf{A}$
$\mathbf{c}$	A column of $\mathbf{A}$
$\mathbf{C}$	A block of d columns of $\mathbf{A}$
$\mathbf{Q}$	The eigenvector matrix for eigendecompsition of $\mathbf{B}^T\mathbf{B}$
$\mathbf{\Lambda}$	The eigenvalue matrix for eigendecomposition of $\mathbf{B}^T\mathbf{B}$

B Proofs and details

B.1 Proof of Theorem 1

We first review the tree structure of the matrix $\mathbf{A}$. $\mathbf{A}$ is an $N \times M$ matrix where every block of d columns corresponds to the effects for one group (e.g., an individual subject, age, school, or gender). For example, if $N = 3$, $k = 2$ and $d = 2$, one possible graphical model is as below.

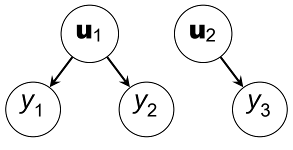

Figure 5: A tree-structured model conditioned on $\mathbf{\Theta}$.

Each $\mathbf{u}_j \in \mathbb{R}^2$. If the coefficients are all 1s, then

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

To generalize, if for y_i, the grouping variable is g_i, then in the ith row of $\mathbf{A}$, only $\mathbf{A}_{i,j:k}$ can be nonzero for $j = (g_i - 1)d + 1$ and $k = g_i d$. We consider three representations of the matrix $\mathbf{A}$. By rows,

$$\mathbf{A} = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \cdots \\ \mathbf{r}_N \end{pmatrix},$$

by columns,

$$\mathbf{A} = \begin{pmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_{kd} \end{pmatrix},$$

and by blocks of d columns,

$$\begin{aligned} \mathbf{A} &= \begin{pmatrix} \mathbf{A}_{:,1:d} & \mathbf{A}_{:,d+1:2d} & \cdots & \mathbf{A}_{:,(k-1)d+1:kd} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{C}_1 & \mathbf{C}_2 & \cdots & \mathbf{C}_k \end{pmatrix} \end{aligned}$$

where each $\mathbf{C}_i$ $(i = 1, 2, \ldots, k)$ is $N \times d$. Now we restate and prove Theorem 1.

Theorem 1. *If $\mathbf{\Sigma_y}$ is diagonal, $\mathbf{\Sigma_u}$ is block-diagonal with blocks of size $d \times d$, then $\mathbf{F} = \mathbf{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T \mathbf{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$ is also block-diagonal with $d \times d$ blocks and computing $\mathbf{A}^T \mathbf{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$ takes $O(Nd^2)$ time.*

Proof. The theorem has two parts: (a) the property of $\mathbf{F}$, and (b) the computation of $\mathbf{F}$. We address them with the three representations of $\mathbf{A}$.

(a) $\mathbf{F}$ is block-diagonal. Because $\mathbf{\Sigma_u}$ is block-diagonal, $\mathbf{\Sigma}_\mathbf{u}^{-1}$ is also block-diagonal with the same sizes. Also, $\mathbf{\Sigma_y}$ is diagonal, so the block-diagonality of $\mathbf{A}^T \mathbf{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$ is the same as $\mathbf{A}^T \mathbf{A}$. We consider the column representation of $\mathbf{A}$, then

$$\begin{aligned} \mathbf{A}^T \mathbf{A} &= \begin{pmatrix} \mathbf{C}_1^T \\ \mathbf{C}_2^T \\ \cdots \\ \mathbf{C}_k^T \end{pmatrix} \begin{pmatrix} \mathbf{C}_1 & \mathbf{C}_2 & \cdots & \mathbf{C}_k \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{C}_1^T \mathbf{C}_1 & \mathbf{C}_1^T \mathbf{C}_2 & \cdots & \mathbf{C}_1^T \mathbf{C}_k \\ \mathbf{C}_2^T \mathbf{C}_1 & \mathbf{C}_2^T \mathbf{C}_2 & \cdots & \mathbf{C}_2^T \mathbf{C}_k \\ \cdots \\ \mathbf{C}_k^T \mathbf{C}_1 & \mathbf{C}_k^T \mathbf{C}_2 & \cdots & \mathbf{C}_k^T \mathbf{C}_k \end{pmatrix}. \end{aligned}$$

For $1 \le i \le k$, $\mathbf{C}_i^T \mathbf{C}_i$ is $d \times d$. For $1 \le i < j \le k$,

$$\mathbf{C}_i^T \mathbf{C}_j = \begin{pmatrix} \mathbf{c}_{(i-1)d+1}^T \\ \mathbf{c}_{(i-1)d+2}^T \\ \cdots \\ \mathbf{c}_{id}^T \end{pmatrix} \begin{pmatrix} \mathbf{c}_{(j-1)d+1} & \mathbf{c}_{(j-1)d+2} & \cdots & \mathbf{c}_{jd} \end{pmatrix}.$$

The following lemma shows that $\mathbf{C}_i^T \mathbf{C}_j = \mathbf{0}$.

Lemma 1. *For any* $1 \le i < j \le k$ *and* $1 \le s, t \le d$, *it holds that* $\mathbf{c}_{(i-1)d+s}^T \mathbf{c}_{(j-1)d+t} = 0$.

Proof. The lemma can be proved by contradiction. Suppose $\mathbf{c}_{(i-1)d+s}^T \mathbf{c}_{(j-1)d+t} \neq 0$ Then there exists an index n such that $\mathbf{c}_{(i-1)d+s}[n] \neq 0$ and $\mathbf{c}_{(j-1)d+t}[n] \neq 0$. This means that in the nth row of $\mathbf{A}$, both $\mathbf{A}_{n,(i-1)d+s}$ and $\mathbf{A}_{n,(j-1)d+t}$ are non-zero. This contradicts with the tree-structure where only one group of d elements can be non-zero in a row. □

With the lemma, we have that $\mathbf{A}^T\mathbf{A}$ is block-diagonal, thus $\mathbf{\Sigma}_\mathbf{u}^{-1} + \mathbf{A}^T \mathbf{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$ is also block-diagonal and each block is $d \times d$.

(b) The computation of $\mathbf{A}^T \mathbf{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$ is $O(Nd^2)$. Since $\mathbf{\Sigma}_\mathbf{y}$ is diagonal, $\mathbf{A}' = \mathbf{\Sigma}_\mathbf{y}^{-1}\mathbf{A}$ has the same pattern of zeros and nonzeros as $\mathbf{A}$. We consider the row representations such that

$$\mathbf{A}' = \begin{pmatrix} \mathbf{r}_1' \\ \mathbf{r}_2' \\ \cdots \\ \mathbf{r}_N' \end{pmatrix}.$$

Then

$$\mathbf{A}^T\mathbf{A}' = \begin{pmatrix} \mathbf{r}_1^T & \mathbf{r}_2^T & \cdots & \mathbf{r}_N^T \end{pmatrix} \begin{pmatrix} \mathbf{r}_1' \\ \mathbf{r}_2' \\ \cdots \\ \mathbf{r}_N' \end{pmatrix} = \sum_{i=1}^{N} \mathbf{r}_i^T \mathbf{r}_i'.$$

note that each of $\mathbf{r}_i$ and $\mathbf{r}_i'$ has d non-zero elements. So computing $\mathbf{A}^T\mathbf{A}'$ is $O(Nd^2)$. □

B.2 Pseudocode for recovery after marginalizing one group of random effects

Algorithm 2 Sampling from $p(\mathbf{u}_i | \mathbf{\Theta}, \mathbf{y}, \mathbf{u}_{-i})$

```
$\mathbf{z} = \mathbf{y} - \sum_{j \neq i} \mathbf{A}_j \mathbf{u}_j - \mathbf{A}_i \mu_i - \mathbf{b}$      ▷Sparse matrix multiplication in $O(NLd)$ time
$\mathbf{G} = \mathbf{A}^T \mathbf{\Sigma}_\mathbf{y}^{-1} \mathbf{A}$      ▷Block diagonal computation in $O(Nd^2)$ time
$\mathbf{F} = \mathbf{\Sigma}_\mathbf{u}^{-1} + \mathbf{G}$      ▷Block diagonal computation in $O(Md^2)$ time
$\mu = \mu_i + \mathbf{\Sigma}_\mathbf{u}(\mathbf{I} - \mathbf{G}\mathbf{F}^{-1})\mathbf{A}^T\mathbf{\Sigma}_\mathbf{y}^{-1}\mathbf{z}$      ▷Sparse matrix multiplication in $O((M+N)d)$ time
$\mathbf{L} =$ Cholesky($\mathbf{F}^{-1}$)      ▷Cholesky of block diagonal matrix in $O(Md^2)$ time
return $\mathbf{u}_i \sim$ Normal($\mu, \mathbf{L}\mathbf{L}^T$).
```

B.3 Details of scaled identity covariance matrices

With the assumptions of scaled identity covariance matrices, all effects can be marginalized with a preprocessing of the eigendecomposition of $\mathbf{B}^T\mathbf{B}$.

(a) Preprocessing before HMC. We compute

$$\mathbf{B}^T\mathbf{B} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T.$$

In LMMs, the computation of $\mathbf{B}^T\mathbf{B}$ is $O(NL^2d^2)$[2], and the eigendecomposition of it is $O(D^3)$. So the overall complexity for preprocessing is $O(D^3 + NL^2d^2)$. Compared with the HMC sampling

[2]Each $\mathbf{A}_i^T\mathbf{A}_j$ is $O(Nd^2)$, as a corollary of Theorem 1.

loop that takes thousands of steps and visits the model hundreds of times each step, the cost of preprocessing is not expensive. In our attempt to marginalize all random effects for the instructor evaluation model in Section 6.1, this step takes less than 10 seconds.

(b) Marginalized likelihood during HMC. During HMC sampling, the log density $\log p(\mathbf{y}|\boldsymbol{\Theta})$ would be calculated, which is

$$\log p(\mathbf{y}|\boldsymbol{\Theta}) = -\frac{1}{2}\det(\mathbf{E}) - \frac{1}{2}\mathbf{z}^T\mathbf{E}^{-1}\mathbf{z} + C$$

where $\mathbf{z} = \mathbf{y} - \mathbf{B}\boldsymbol{\mu} - \mathbf{b}$ and $\mathbf{E} = \mathbf{B}\boldsymbol{\Sigma}_{\mathbf{v}}\mathbf{B}^T + \boldsymbol{\Sigma}_{\mathbf{y}}$. The computation of $\mathbf{z}$ takes $O(NLd)$ time. With the two lemmas, we have

$$\det(\mathbf{E}) = \det(\boldsymbol{\Sigma}_{\mathbf{v}}^{-1} + \mathbf{B}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{B})\det(\boldsymbol{\Sigma}_{\mathbf{v}})\det(\boldsymbol{\Sigma}_{\mathbf{y}}),$$
$$\mathbf{z}^T\mathbf{E}^{-1}\mathbf{z} = \mathbf{z}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{z} - \mathbf{z}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{B}(\boldsymbol{\Sigma}_{\mathbf{v}}^{-1} + \mathbf{B}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{z}.$$

A shared matrix in the formulas is $\mathbf{F} = \boldsymbol{\Sigma}_{\mathbf{v}}^{-1} + \mathbf{B}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{B}$. Then

$$\mathbf{F} = \boldsymbol{\Sigma}_{\mathbf{v}}^{-1} + \mathbf{B}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{B} = \mathbf{Q}\left(\frac{1}{\tau_{\mathbf{v}}}\mathbf{I} + \frac{1}{\tau_{\mathbf{y}}}\boldsymbol{\Lambda}\right)\mathbf{Q}^T.$$

With the trick, evaluating $\det(\mathbf{F})$ reduced to $O(D)$ time as $\det(\mathbf{Q}) = 1$. Also $\mathbf{z}^T\mathbf{E}^{-1}\mathbf{z}$ becomes

$$\mathbf{z}^T\mathbf{E}^{-1}\mathbf{z} = \mathbf{z}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{z} - \mathbf{z}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{B}\mathbf{Q}\left(\frac{1}{\tau_{\mathbf{v}}}\mathbf{I} + \frac{1}{\tau_{\mathbf{y}}}\boldsymbol{\Lambda}\right)^{-1}\mathbf{Q}^T\mathbf{B}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{z}.$$

Note that $\mathbf{B}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{z} \in \mathrm{R}^D$ can be computed in $O(NLd)$ time, but its multiplication with $\mathbf{Q}^T$ takes $O(D^2)$ time. Given that $\frac{1}{\tau_{\mathbf{v}}}\mathbf{I} + \frac{1}{\tau_{\mathbf{y}}}\boldsymbol{\Lambda}$ is diagonal, the complexity of evaluating $\log p(\mathbf{y}|\boldsymbol{\Theta})$ once is then $O(D^2 + NLd)$.

(c) Ancestral sampling after HMC. In the recovery step, we perform ancestral sampling from $p(\mathbf{v}|\boldsymbol{\Theta}, \mathbf{y})$. To efficiently generate samples, we give the following theorem.

Theorem 2. *If* $\boldsymbol{\Sigma}_{\mathbf{v}} = \tau_{\mathbf{v}}\mathbf{I}$, $\boldsymbol{\Sigma}_{\mathbf{y}} = \tau_{\mathbf{y}}\mathbf{I}$ *and* $\mathbf{B}^T\mathbf{B} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$, *then*

$$\mathbf{v}|\boldsymbol{\Theta}, \mathbf{y} \sim N(\boldsymbol{\mu}_{\mathbf{v}|\boldsymbol{\Theta},\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{v}|\boldsymbol{\Theta},\mathbf{y}}),$$

where

$$\boldsymbol{\mu}_{\mathbf{v}|\boldsymbol{\Theta},\mathbf{y}} = \boldsymbol{\mu} + \frac{\tau_{\mathbf{v}}}{\tau_{\mathbf{y}}}\left(\mathbf{B}^T - \frac{1}{\tau_{\mathbf{y}}}\mathbf{Q}\boldsymbol{\Lambda}\left(\frac{1}{\tau_{\mathbf{v}}} + \frac{\boldsymbol{\Lambda}}{\tau_{\mathbf{y}}}\right)^{-1}\mathbf{Q}^T\mathbf{B}^T\right)\mathbf{z},$$
$$\boldsymbol{\Sigma}_{\mathbf{v}|\boldsymbol{\Theta},\mathbf{y}} = \mathbf{Q}\left(\frac{1}{\tau_{\mathbf{v}}}\mathbf{I} + \frac{1}{\tau_{\mathbf{y}}}\boldsymbol{\Lambda}\right)^{-1}\mathbf{Q}^T.$$

In Theorem 2, from $\mathbf{z}$, we can apply matrix multiplications from right to left to get $\boldsymbol{\mu}_{\mathbf{v}|\boldsymbol{\Theta},\mathbf{y}}$. The whole computation takes $O(D^2 + NLd)$. To generate normal samples a Cholseky factorization for $\boldsymbol{\Sigma}_{\mathbf{v}|\boldsymbol{\Theta},\mathbf{y}}$ is required. But $\left(\frac{1}{\tau_{\mathbf{v}}}\mathbf{I} + \frac{1}{\tau_{\mathbf{y}}}\boldsymbol{\Lambda}\right)^{-1}$ is diagonal, so it can be obtained in $O(D^2)$ time as well. Now we prove Theorem 2.

Proof. $\boldsymbol{\mu}_{\mathbf{v}|\boldsymbol{\Theta},\mathbf{y}}$ and $\boldsymbol{\Sigma}_{\mathbf{v}|\boldsymbol{\Theta},\mathbf{y}}$ can both be derived algebraically.

$$\begin{aligned}
\boldsymbol{\mu}_{\mathbf{v}|\boldsymbol{\Theta},\mathbf{y}} &= \boldsymbol{\mu} + \mathbf{M}\mathbf{z} \\
&= \boldsymbol{\mu} + \boldsymbol{\Sigma}_{\mathbf{v}}\mathbf{B}^T(\mathbf{B}\boldsymbol{\Sigma}_{\mathbf{v}}\mathbf{B}^T + \boldsymbol{\Sigma}_{\mathbf{y}})^{-1}\mathbf{z} \\
&= \boldsymbol{\mu} + \boldsymbol{\Sigma}_{\mathbf{v}}\mathbf{B}^T(\boldsymbol{\Sigma}_{\mathbf{y}}^{-1} - \boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{B}(\boldsymbol{\Sigma}_{\mathbf{v}}^{-1} + \mathbf{B}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1})\mathbf{z} \\
&= \boldsymbol{\mu} + \frac{\tau_{\mathbf{v}}}{\tau_{\mathbf{y}}}(\mathbf{B}^T - \frac{1}{\tau_{\mathbf{y}}}\mathbf{B}^T\mathbf{B}(\boldsymbol{\Sigma}_{\mathbf{v}}^{-1} + \mathbf{B}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{B})^{-1}\mathbf{B}^T)\mathbf{z} \\
&= \boldsymbol{\mu} + \frac{\tau_{\mathbf{v}}}{\tau_{\mathbf{y}}}(\mathbf{B}^T - \frac{1}{\tau_{\mathbf{y}}}\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T\mathbf{Q}\left(\frac{1}{\tau_{\mathbf{v}}}\mathbf{I} + \frac{1}{\tau_{\mathbf{y}}}\boldsymbol{\Lambda}\right)^{-1}\mathbf{Q}^T\mathbf{B}^T)\mathbf{z} \\
&= \boldsymbol{\mu} + \frac{\tau_{\mathbf{v}}}{\tau_{\mathbf{y}}}(\mathbf{B}^T - \frac{1}{\tau_{\mathbf{y}}}\mathbf{Q}\boldsymbol{\Lambda}\left(\frac{1}{\tau_{\mathbf{v}}}\mathbf{I} + \frac{1}{\tau_{\mathbf{y}}}\boldsymbol{\Lambda}\right)^{-1}\mathbf{Q}^T\mathbf{B}^T)\mathbf{z}.
\end{aligned}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{\mathbf{v}|\boldsymbol{\Theta},\mathbf{y}} &= (\mathbf{I} - \mathbf{MB})\boldsymbol{\Sigma}_{\mathbf{v}} \\ &= (\mathbf{I} - \boldsymbol{\Sigma}_{\mathbf{v}}\mathbf{B}^{T}(\mathbf{B}\boldsymbol{\Sigma}_{\mathbf{v}}\mathbf{B}^{T} + \boldsymbol{\Sigma}_{\mathbf{y}})^{-1}\mathbf{B})\boldsymbol{\Sigma}_{\mathbf{v}} \\ &= (\boldsymbol{\Sigma}_{\mathbf{v}}^{-\mathbf{1}} + \mathbf{B}^{T}\boldsymbol{\Sigma}_{\mathbf{y}}^{-\mathbf{1}}\mathbf{B})^{-1} \\ &= \mathbf{Q}\left(\frac{1}{\tau_{\mathbf{v}}}\mathbf{I} + \frac{1}{\tau_{\mathbf{y}}}\boldsymbol{\Lambda}\right)^{-1}\mathbf{Q}^{T}.\end{aligned}$$

$\square$

C Additional experimental results

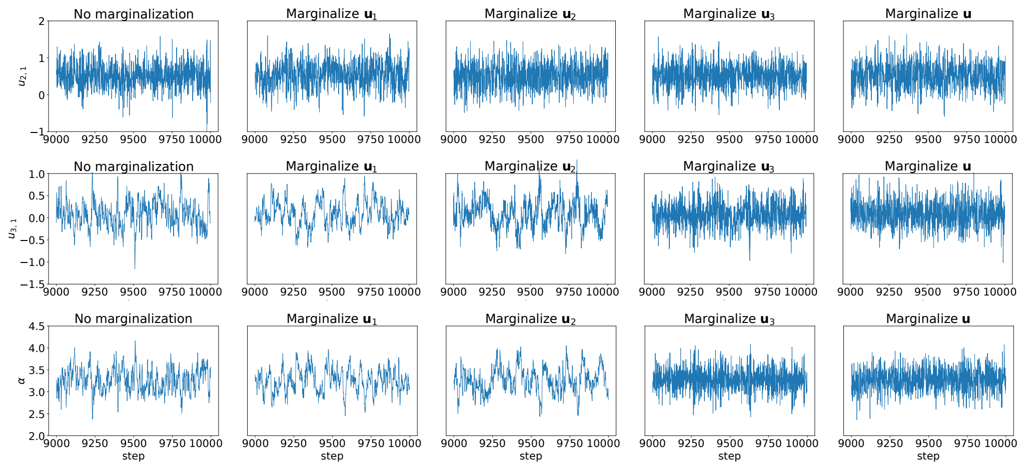

Figure 6: Trace plots for $u_{2,1}$, $u_{3,1}$ and α of an interval of 1,000 sampling steps after warmup on the ETH instructor evaluation model, using the same data as Figure 2 in the paper.

Table 5: Number of parameters (out of 4117) whose $\hat{R}$ exceed a threshold for 1,000 samples from HMC, with or without marginalization. Mean and standard deviation over 5 independent runs are reported.

Threshold	No marginalization	Marginalize $\mathbf{u}_1$	Marginalize $\mathbf{u}_2$	Marginalize $\mathbf{u}_3$	Marginalize $\mathbf{u}$
> 1.01	186.80 (37.26)	295.60 (134.57)	11.80 (6.05)	59.80 (37.35)	5.20 (1.72)
> 1.02	99.40 (18.91)	153.20 (99.90)	6.20 (7.19)	9.80 (10.48)	0.00 (0.00)
> 1.05	13.40 (6.83)	54.00 (51.99)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
> 1.10	0.00 (0.00)	23.20 (24.51)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)

Table 6: Divergence (mean and standard deviation) out of 10,000 samples with different strategies on the grouseticks model across 5 random seeds under different target probabilities. We use M1 to represent marginalizing $\mathbf{u}_1$, M2 to represent marginalizing $\mathbf{u}_2$, R1 to represent reparameterizing $\mathbf{u}_1$, R2 to represent reparameterizing $\mathbf{u}_2$.

Transformation	Number of divergence
No marginalization	42.60 (25.76)
M1	14.60 (13.85)
M2	0.00 (0.00)
M1, R2	22.60 (18.13)
M2, R1	0.00 (0.00)
R1, R2	431.60 (507.37)

D Models and example probabilistic programs

We provide the details of the nine cognitive science datasets and their corresponding models and probabilistic programs. We follow [47] and use maximal models with correlated varying intercept and slopes for each of the datasets. The model for the pupil dataset is described in Section 2.

D.1 Agreement attraction in comprehension

The dataset (dillonE1) studies the effect of the agreement attraction phenomenon when reading a noun with the auxiliary verb [12]. The predictor is

$$\log(y_i) = \alpha + u_{1,g_{1,i},1} + u_{2,g_{2,i},1} + t_i(\beta + u_{1,g_{1,i},2} + u_{2,g_{2,i},2}) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2).$$

Each experiment result y_i is from subject $g_{1,i}$ on sentence $g_{2,i}$, with t_i being the interference level ($t_i \in \{0, 1\}$). Bayesian hierarchical modeling assigns prior to the variables.

$$\mathbf{T}_1 \sim \mathcal{N}^+(\mathbf{0}, \text{diag}(5^2, 5^2)),\ \mathbf{L}_1 \sim \text{LKJCholesky}(2, 1),\ \mathbf{T}_2 \sim \mathcal{N}^+(\mathbf{0}, \text{diag}(5^2, 5^2)),\ \mathbf{L}_2 \sim \text{LKJCholesky}(2, 1),$$
$$\alpha \sim \mathcal{N}(0, 10^2),\ \beta \sim \mathcal{N}(0, 5^2),\ \sigma \sim \mathcal{N}^+(0, 5^2),\ \mathbf{u}_{1,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{T}_1\mathbf{L}_1\mathbf{L}_1^T\mathbf{T}_1),\ \mathbf{u}_{2,k} \sim \mathcal{N}(\mathbf{0}, \mathbf{T}_2\mathbf{L}_2\mathbf{L}_2^T\mathbf{T}_2).$$

The probabilistic program in NumPyro is then

```
def model(n_sub, n_item, n_obs, g1, g2, treatment, obs):
    alpha = numpyro.sample('alpha', dist.Normal(0, 10))
    beta = numpyro.sample('beta', dist.Normal(0, 5))
    sigma = numpyro.sample('sigma', dist.HalfNormal(5))
    sigma_u = numpyro.sample('sigma_u', dist.LKJCholesky(2))
    tau_u = numpyro.sample('tau_u', dist.HalfNormal(5), sample_shape=(2, ))
    sigma_v = numpyro.sample('sigma_v', dist.LKJCholesky(2))
    tau_v = numpyro.sample('tau_v', dist.HalfNormal(5), sample_shape=(2, ))
    s_u = jnp.matmul(jnp.diag(tau_u), sigma_u)
    s_v = jnp.matmul(jnp.diag(tau_v), sigma_v)
    u = numpyro.sample('u', dist.MultivariateNormal(jnp.zeros((2,)), scale_tril=s_u), sample_shape=(n_sub,))
    v = numpyro.sample('v', dist.MultivariateNormal(jnp.zeros((2,)),scale_tril=s_v), sample_shape=(n_item,))
    numpyro.sample('y', dist.LogNormal(alpha + u[g1][...,0] + v[g2][...,0] +
                                       treatment * (beta + u[g1][...,1] + v[g2][...,1]), sigma), obs=obs)
```

We use u and v in the codes to represent the two random effects. The probabilistic program with marginalization is similar. Suppose we marginalize u, our probabilistic program becomes

```
def model(n_sub, n_item, n_obs, g1, g2, treatment, obs):
    alpha = numpyro.sample('alpha', dist.Normal(0, 10))
    beta = numpyro.sample('beta', dist.Normal(0, 5))
    sigma = numpyro.sample('sigma', dist.HalfNormal(5))
    sigma_u = numpyro.sample('sigma_u', dist.LKJCholesky(2))
    tau_u = numpyro.sample('tau_u', dist.HalfNormal(5), sample_shape=(2, ))
    sigma_v = numpyro.sample('sigma_v', dist.LKJCholesky(2))
    tau_v = numpyro.sample('tau_v', dist.HalfNormal(5), sample_shape=(2, ))
    s_u = jnp.matmul(jnp.diag(tau_u), sigma_u)
    s_v = jnp.matmul(jnp.diag(tau_v), sigma_v)
    u = jnp.zeros((n_sub, 2))
    v = numpyro.sample('v', dist.MultivariateNormal(jnp.zeros((2,)),scale_tril=s_v), sample_shape=(n_item,))
    numpyro.sample('y', MarginalizedLogNormalGroupCoeff(alpha + u[g1][...,0] + v[g2][...,0] +
                        treatment * (beta + u[g1][...,1] + v[g2][...,1]), s_u, sigma, g1, treatment, n_sub, n_obs, u), obs=obs)
```

To marginalize v, the probabilistic program is

```
def model(n_sub, n_item, n_obs, g1, g2, treatment, obs):
    alpha = numpyro.sample('alpha', dist.Normal(0, 10))
    beta = numpyro.sample('beta', dist.Normal(0, 5))
    sigma = numpyro.sample('sigma', dist.HalfNormal(5))
    sigma_u = numpyro.sample('sigma_u', dist.LKJCholesky(2))
    tau_u = numpyro.sample('tau_u', dist.HalfNormal(5), sample_shape=(2, ))
    sigma_v = numpyro.sample('sigma_v', dist.LKJCholesky(2))
    tau_v = numpyro.sample('tau_v', dist.HalfNormal(5), sample_shape=(2, ))
    s_u = jnp.matmul(jnp.diag(tau_u), sigma_u)
    s_v = jnp.matmul(jnp.diag(tau_v), sigma_v)
    v = jnp.zeros((n_item, 2))
    u = numpyro.sample('u', dist.MultivariateNormal(jnp.zeros((2,)), scale_tril=s_u), sample_shape=(n_sub,))
    numpyro.sample('y', MarginalizedMultivariateLogNormalGroupCoeff(alpha + u[g1][...,0] + v[g2][...,0] +
                        treatment * (beta + u[g1][...,1] + v[g2][...,1]), s_v, sigma, g2, treatment, n_item, n_obs, v), obs=obs)
```

The probabilistic programs for the other models will be similar and we omit them for simplicity.

D.2 English and Dutch Grammaticality illusion

The datasets (english [69], dutch [17]) study the VP-forgetting hypothesis [25] for different languages. They use the same predictor and priors. The predictor is

$$y_i = \alpha + u_{1,g_{1,i},1} + u_{2,g_{2,i},1} + t_i(\beta + u_{1,g_{1,i},2} + u_{2,g_{2,i},2}) + \epsilon,\ \epsilon \sim N(0, \sigma^2),$$

where t_i is the treatment variable and $t_i \in \{-1, 1\}$. And the prior is

$$\mathbf{T}_1 \sim N^+(\mathbf{0}, \mathrm{diag}(1^2, 1^2)),\ \mathbf{L}_1 \sim \mathrm{LKJCholesky}(2, 1),\ \mathbf{T}_2 \sim N^+(\mathbf{0}, \mathrm{diag}(1^2, 1^2)),\ \mathbf{L}_2 \sim \mathrm{LKJCholesky}(2, 1),$$
$$\alpha \sim N(0, 10^2),\ \beta \sim N(0, 5^2),\ \sigma \sim N^+(0, 5^2),\ \mathbf{u}_{1,j} \sim N(\mathbf{0}, \mathbf{T}_1\mathbf{L}_1\mathbf{L}_1^T\mathbf{T}_1),\ \mathbf{u}_{2,k} \sim N(\mathbf{0}, \mathbf{T}_2\mathbf{L}_2\mathbf{L}_2^T\mathbf{T}_2).$$

D.3 Electrophysiological responses with N400 effect

In the study of language, the electroencephalography (EGG) responses with N400 effect is studied [47]. Experimental results of subjects from the Edinburgh lab are collected [48]. The predictor is

$$y_i = \alpha + u_{1,g_{1,i},1} + u_{2,g_{2,i},1} + t_i(\beta + u_{1,g_{1,i},2} + u_{2,g_{2,i},2}) + \epsilon,\ \epsilon \sim N(0, \sigma^2),$$

where t_i is the treatment variable and $t_i \in [0, 1]$. And the prior is

$$\mathbf{T}_1 \sim N^+(\mathbf{0}, \mathrm{diag}(20^2, 20^2)),\ \mathbf{L}_1 \sim \mathrm{LKJCholesky}(2, 1),\ \mathbf{T}_2 \sim N^+(\mathbf{0}, \mathrm{diag}(20^2, 20^2)),\ \mathbf{L}_2 \sim \mathrm{LKJCholesky}(2, 1),$$
$$\alpha \sim N(0, 10^2),\ \beta \sim N(0, 10^2),\ \sigma \sim N^+(0, 50^2),\ \mathbf{u}_{1,j} \sim N(\mathbf{0}, \mathbf{T}_1\mathbf{L}_1\mathbf{L}_1^T\mathbf{T}_1),\ \mathbf{u}_{2,k} \sim N(\mathbf{0}, \mathbf{T}_2\mathbf{L}_2\mathbf{L}_2^T\mathbf{T}_2).$$

D.4 Subjective and objective relatives

Grodner and Gibson [29] (gg05) studies the processing time difference between object relative clause and subject relative clause sentences. The predictor is

$$\log(y_i) = \alpha + u_{1,g_{1,i},1} + u_{2,g_{2,i},1} + u_{3,g_{3,i},1} + t_i(\beta + u_{1,g_{1,i},2} + u_{2,g_{2,i},2} + u_{3,g_{3,i},1}) + \epsilon,\ \epsilon \sim N(0, \sigma^2),$$

and the treatment variable $t_i \in \{-1, 1\}$. The third effect $\mathbf{u}_3$ is related to different repeats of the experiment and has only two groups. We consider the first two effects for marginalization to match the other experiments. The prior for the variables is

$$\mathbf{T}_1 \sim N^+(\mathbf{0}, \mathrm{diag}(5^2, 5^2)),\ \mathbf{T}_2 \sim N^+(\mathbf{0}, \mathrm{diag}(5^2, 5^2)),\ \mathbf{T}_3 \sim N^+(\mathbf{0}, \mathrm{diag}(5^2, 5^2)),$$
$$\mathbf{L}_1 \sim \mathrm{LKJCholesky}(2, 1),\ \mathbf{L}_2 \sim \mathrm{LKJCholesky}(2, 1),\ \mathbf{L}_3 \sim \mathrm{LKJCholesky}(2, 1),$$
$$\alpha \sim N(0, 10^2),\ \beta \sim N(0, 5^2),\ \sigma \sim N^+(0, 5^2),$$
$$\mathbf{u}_{1,j} \sim N(\mathbf{0}, \mathbf{T}_1\mathbf{L}_1\mathbf{L}_1^T\mathbf{T}_1),\ \mathbf{u}_{2,k} \sim N(\mathbf{0}, \mathbf{T}_2\mathbf{L}_2\mathbf{L}_2^T\mathbf{T}_2),\ \mathbf{u}_{3,l} \sim N(\mathbf{0}, \mathbf{T}_3\mathbf{L}_3\mathbf{L}_3^T\mathbf{T}_3).$$

D.5 Relative clause processing in Mandarin Chinese

The datasets (mandarin [75], mandarin2 [70]) are collected from experiments to study the effect of relative clause type on reading time of Mandarin Chinese. In our model, the predictor is

$$\log(y_i) = \alpha + u_{1,g_{1,i},1} + u_{2,g_{2,i},1} + t_i(\beta + u_{1,g_{1,i},2} + u_{2,g_{2,i},2}) + \epsilon,\ \epsilon \sim N(0, \sigma^2),$$

where t_i is the treatment variable and $t_i \in \{-0.5, 0.5\}$. And the prior is

$$\mathbf{T}_1 \sim N^+(\mathbf{0}, \mathrm{diag}(5^2, 5^2)),\ \mathbf{L}_1 \sim \mathrm{LKJCholesky}(2, 1),\ \mathbf{T}_2 \sim N^+(\mathbf{0}, \mathrm{diag}(5^2, 5^2)),\ \mathbf{L}_2 \sim \mathrm{LKJCholesky}(2, 1),$$
$$\alpha \sim N(0, 10^2),\ \beta \sim N(0, 5^2),\ \sigma \sim N^+(0, 5^2),\ \mathbf{u}_{1,j} \sim N(\mathbf{0}, \mathbf{T}_1\mathbf{L}_1\mathbf{L}_1^T\mathbf{T}_1),\ \mathbf{u}_{2,k} \sim N(\mathbf{0}, \mathbf{T}_2\mathbf{L}_2\mathbf{L}_2^T\mathbf{T}_2).$$

D.6 The Stroop effect

The Stroop effect describes the change of response time between congruent and incongruent stimuli [38]. The dataset is from Ebersole et al. [16]. Different from the other models, the noise scale for each observation is also grouped. In our model, the predictor is

$$\log(y_i) = \alpha + u_{g_i,1} + t_i(\beta + u_{g_i,2}) + \epsilon,\ \ \epsilon \sim N(0, \sigma_i^2),\ \ \sigma_i = \exp(\sigma_\alpha + s_{g_i,1} + t_i(\sigma_\beta + s_{g_i,2})),$$

and the treatment variable is $t_i \in \{-1, 1\}$. Priors for the model are

$$\mathbf{T}_\mathbf{u} \sim N^+(\mathbf{0}, \mathrm{diag}(1, 1)),\ \mathbf{L}_\mathbf{u} \sim \mathrm{LKJCholesky}(2, 1),\ \mathbf{T}_\sigma \sim N^+(\mathbf{0}, \mathrm{diag}(1, 1)),\ \mathbf{L}_\sigma \sim \mathrm{LKJCholesky}(2, 1),$$
$$\alpha \sim N(6, 1.5^2),\ \beta \sim N(0, 0.01^2),\ \sigma_\alpha \sim N(0, 1),\ \sigma_\beta \sim N(0, 1),$$
$$\mathbf{u}_j \sim N(\mathbf{0}, \mathbf{T}_\mathbf{u}\mathbf{L}_\mathbf{u}\mathbf{L}_\mathbf{u}^T\mathbf{T}_\mathbf{u}),\ \mathbf{s}_j \sim N(\mathbf{0}, \mathbf{T}_\sigma\mathbf{L}_\sigma\mathbf{L}_\sigma^T\mathbf{T}_\sigma).$$