# Is Gibbs sampling faster than Hamiltonian Monte Carlo on GLMs?

**Son Luu**
University of British Columbia

**Zuheng Xu**
University of British Columbia

**Nikola Surjanovic**
University of British Columbia

**Miguel Biron-Lattes**
University of British Columbia

**Trevor Campbell**
University of British Columbia

**Alexandre Bouchard-Côté**
University of British Columbia

## Abstract

The Hamiltonian Monte Carlo (HMC) algorithm is often lauded for its ability to effectively sample from high-dimensional distributions. In this paper we challenge the presumed domination of HMC for the Bayesian analysis of GLMs. By utilizing the structure of the compute graph rather than the graphical model, we show a reduction of the time per sweep of a full-scan Gibbs sampler from $O(d^2)$ to $O(d)$, where $d$ is the number of GLM parameters. A simple change to the implementation of the Gibbs sampler allows us to perform Bayesian inference on high-dimensional GLMs that are practically infeasible with traditional Gibbs sampler implementations. We empirically demonstrate a substantial increase in effective sample size per time when comparing our Gibbs algorithms to state-of-the-art HMC algorithms. While Gibbs is superior in terms of dimension scaling, neither Gibbs nor HMC dominate the other: we provide numerical and theoretical evidence that HMC retains an edge in certain circumstances thanks to its advantageous condition number scaling. Interestingly, for GLMs of fixed data size, we observe that increasing dimensionality can stabilize or even decrease condition number, shedding light on the empirical advantage of our efficient Gibbs sampler.

## 1 INTRODUCTION

Generalized linear models (GLMs) are among the most commonly used tools in contemporary Bayesian statistics (Gelman et al., 2013). As a result, applied Bayesian statisticians require efficient algorithms to approximate the posterior distributions associated with GLM parameters, often relying on Markov chain Monte Carlo (MCMC). In this paper we focus on two MCMC samplers: the Gibbs sampler and Hamiltonian Monte Carlo (HMC), popularized in statistics by Geman and Geman (1984) and Neal (1996), respectively.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

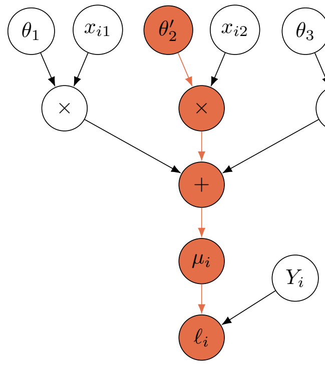


Figure 1: Part of a GLM compute graph with three regression parameters. In the figure we compute $\ell_i$, the log-likelihood corresponding to data point $i$. Using caching techniques we reduce the complexity of Gibbs from $O(d^2)$ to $O(d)$. Our set of regression parameters after a coordinate update on $j = 2$ with a Gibbs sampler is $\theta' = (\theta_1, \theta'_2, \theta_3)$. To recompute $\ell_i$ after updating $\theta'_2$, we only need to modify the entries highlighted in red. Using caches updated by subtracting the old value and adding the new one at the '+' node, this change can be performed in $O(1)$ operations, instead of $O(d)$ operations. Over $d$ coordinates, Gibbs then has a cost of $O(d)$ instead of $O(d^2)$.

Until around 2010, Gibbs sampling was the predominant option, as it formed the core of a "first generation" of probabilistic programming languages (PPLs) (Štrumbelj et al., 2024) such as BUGS (Lunn et al., 2009) and JAGS (Plummer et al., 2003). A large shift occurred in the early 2010s when HMC and its variant, the No-U-Turn Sampler (NUTS) (Hoffman and Gelman, 2014), were combined with reverse-mode automatic differentiation to power a "second generation" of PPLs such as Stan (Carpenter et al., 2017). Subsequently, HMC largely replaced Gibbs as the focus of attention of the methodological, theoretical, and applied MCMC communities. A notable exception is Gibbs sampling under specific linear and conditionally Gaussian models (Papaspiliopoulos et al., 2020; Zanella and Roberts, 2021).

Our first contribution is the analysis of an algorithm for Gibbs sampling that speeds up inference of GLMs by a factor $O(d)$, where $d$ is the number of parameters, compared to the Gibbs implementation used by "first generation" PPLs. The main idea behind this speedup is to use information encoded in the *compute graph* associated with the target log density—in contrast, previous Gibbs algorithms relied on graphical models (Jordan, 2004), which we show is a sub-optimal representation of GLMs for the purpose of efficient Gibbs sampling. The compute graph is a directed acyclic graph (DAG) encoding the dependencies between the operations involved in computing a function. See Figure 1 for an example. Programming strategies to extract and manipulate such graphs are well developed, in large part because compute graphs also play a key role in reverse-mode automatic differentiation (see Margossian (2019) for a review).

This drastic $O(d)$ speedup prompted us to reappraise Gibbs sampling as a method of choice for approximating GLM posterior distributions. A holistic comparison of sampling algorithms requires taking into account not only the running time per iteration, but also the number of iterations to achieve one effective sample (Flegal et al., 2008). To understand the latter, we use a combination of empirical results and theory. On the theoretical side, we consider normal models, where both HMC and Gibbs have well-developed results quantifying the number of iterations needed to achieve one effective sample (Roberts and Sahu, 1997; Ascolani et al., 2024a; Beskos et al., 2013; Chen and Gatmiry, 2023). See Section 4 for the rationale behind the choice of a normal model and an extended bibliography. In the normal setting, the $O(d)$ runtime of our improved Gibbs sampler combined with past theory yields a favourable $O(d)$ floating point operations (flops) per effective sample, compared with the higher $O(d^{5/4})$ flops per effective sample for HMC. While the $d^{1/4}$ advantage of Gibbs over HMC may seem minor, a key point is that Gibbs requires no adaptation to achieve this performance, while HMC needs to be finely tuned adaptively (Livingstone and Zanella, 2022), implying that the practical rate for HMC may be even higher. Note also that previous, graphical model-based implementations of Gibbs sampling require $O(d^2)$ flops per effective sample in the same setup.

Examining scaling in $d$ alone does not provide a full story, even in the normal setting; the shape of the posterior contour lines also has a strong effect on the sampling performance of both Gibbs and HMC. Various notions of *condition number* $\kappa$ are used (Langmore et al., 2019; Hird and Livingstone, 2023) to summarize the complexity brought by the shape of log-concave distributions. In gradient-based methods, a popular notion of condition number is the square of the broadest direction's scale to that of the most constrained. For optimally-tuned HMC with non-constant integration time on Gaussian targets, the cost per effective sample is $O(d^{5/4}\kappa^{1/2})$ flops (Langmore et al., 2019; Apers et al., 2022; Wang and Wibisono, 2023; Jiang, 2023). Understanding the condition number scaling of Gibbs is more nuanced, as Gibbs sampling is invariant to axis-aligned stretching (Román et al., 2014), but is sensitive to rotations (the situation is reversed for HMC (Neal, 2011)). We develop a notion of *residual condition number*, $\kappa_\mathrm{r}$, to capture how Gibbs perceives the target's shape (see Section 2.4). Crucially, $\kappa_\mathrm{r} \leq \kappa$. On the other hand, due to random walk behaviour, Gibbs' cost per effective sample scales as $O(d\kappa_\mathrm{r})$ flops, and in some cases, $\kappa = \kappa_\mathrm{r}$, leading to a potentially worse or better scaling in the condition number for Gibbs compared to HMC, depending on the nature of the target. As a result, neither Gibbs sampling nor HMC dominate each other.

The relative importance of $\kappa$ and $d$ will in general be problem-dependent. To get some insight on how these quantities relate to each other in practice, we investigate sequences of posterior distributions obtained by subsampling an increasing number of covariates from real datasets. We find that various notions of condition number increase until $d \approx n$, after which they either stabilize or, surprisingly, can in certain cases decrease with $d$. This suggests that our efficient Gibbs sampling will be particularly effective in GLMs where $d \gg n$.

We implemented our new Gibbs algorithm and benchmarked it using a collection of synthetic and real datasets. In 11 out of the 12 datasets considered, our algorithm achieves a higher effective sample size (ESS) per second compared to Stan, with a speedup of up to a factor of 300.

## 2 BACKGROUND

In this section we introduce several key concepts for studying the Gibbs sampler and HMC when applied to GLMs. Throughout this paper, we denote the target distribution of interest on $\mathbb{R}^d$ by $\pi$, with density $\pi(\theta) \propto e^{-U(\theta)}$, for $\theta \in \mathbb{R}^d$, with respect to Lebesgue measure on $\mathbb{R}^d$. We assume that for all $\theta \in \mathbb{R}^d$, $\nabla U(\theta)$ and $\nabla^2 U(\theta)$ are well-defined. The distribution $\pi$ is called *strongly log-concave* if there exists a constant $a > 0$ such that for all $\theta \in \mathbb{R}^d$, $\nabla^2 U(\theta) \succeq a I_d$, and *strongly log-smooth* if there exists a constant $b > 0$ such that for all $\theta \in \mathbb{R}^d$, $\|\nabla^2 U(\theta)\| \le b < \infty$.

### 2.1 Gibbs sampling

In the literature, the term "Gibbs sampler" is associated with two key ideas: (1) moving a subset of coordinates while fixing the others; and (2), using the conditional distribution of the target distribution to perform such an update. The algorithm in Section 3 is described for simplicity in the context of Gibbs samplers, but applies more generally to any "*Metropolis-within-Gibbs*" (Chib and Greenberg, 1995) sampler which are those based on (1) only. In the experiments, we use "slice sampling within Gibbs", specifically, with doubling and shrinking (Neal, 2003). See Appendix C.4 for more discussion on Gibbs versus "within-Gibbs."

For $\theta = (\theta_1, \ldots, \theta_d)$ and $j \in \{1, 2, \ldots, d\}$, let $\theta_{-j}$ be the vector containing all components of $\theta$ except for $\theta_j$. We define the conditional distributions $\pi_{j|-j}$, which correspond to the conditional distributions of $\theta_j$ given $\theta_{-j}$, where $\theta \sim \pi$. In Metropolis-within-Gibbs, each coordinate update is performed using a Markov kernel $K_{j|-j}$ that leaves $\pi_{j|-j}$ invariant. The *Gibbs sampler* uses the Markov kernels $K_{j|-j}(x, \cdot) = \pi_{j|-j}(\cdot)$, but more general kernels are commonly used as well (Neal, 2003). It remains to decide the order in which to update coordinates. One popular approach is to use a deterministic update Gibbs sampler (DUGS) (Roberts and Sahu, 1997; Greenwood et al., 1998), where coordinates are updated in a fixed order. For example, when $d = 3$, the DUGS kernel is: $\theta'_1 \sim K_{1|-1}(\cdot|\theta_2, \theta_3)$, followed by $\theta'_2 \sim K_{2|-2}(\cdot|\theta'_1, \theta_3)$, and finally, $\theta'_3 \sim K_{3|-3}(\cdot|\theta'_1, \theta'_2)$. We write $K_{\text{DUGS}}$ for the deterministic alternation of the $d$ Gibbs kernels $K_{j|-j}(x, \cdot) = \pi_{j|-j}(\cdot)$. Options beyond DUGS include the *random scan* Gibbs sampler, which chooses coordinates to update randomly, and *block* Gibbs samplers, which chooses several coordinates to update at once and can improve convergence (Li and Geng, 2005). In this work, we focus primarily on $K_{\text{DUGS}}$, as it yields a smaller asymptotic variance than random scan Gibbs in some problem classes (Greenwood et al., 1998; Qin and Jones, 2022; Andrieu, 2016)—although a definitive conclusion is more nuanced (Roberts and Rosenthal, 2015; He et al., 2016). Note also that random scan Gibbs would yield similar scaling results for the targets we consider (Ascolani et al., 2024a) (see also Appendix E for an empirical demonstration).

### 2.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) is an MCMC algorithm that makes use of gradient information of the target density and introduces a momentum in the sampling process to provide efficient exploration of the state space (Neal, 2011). Calculating the trajectory of HMC samples requires solving a differential equation, which is approximated using numerical integrators in practice. We introduce HMC in two phases: using idealized trajectories, which assume that exact numerical integrators exists; and with the leapfrog integrator, a popular numerical integrator in the case of HMC. In both cases, the HMC kernel has an invariant distribution on an augmented space with momentum variables $p \in \mathbb{R}^d$, with joint density $\bar{\pi}(\theta, p) = \pi(\theta) \cdot \mathcal{N}(p \mid 0, M)$, which admits $\pi$ as the $\theta$-marginal. Here, $M$ is the covariance matrix of the Gaussian momentum, which is often referred to as the *mass matrix* of HMC.

For *idealized HMC*, we initialize with a starting point $\theta^{(0)} = \theta(0)$ at time $t = 0$ and some momentum $p(0) \sim \mathcal{N}(0, M)$. We specify a (possibly random) integration time $\tau$ and solve for $\{\theta(t)\}_{t \in [0,\tau]}$ from

$$\theta'(t) = M^{-1} \cdot p(t), \qquad p'(t) = -\nabla U(\theta(t)), \quad (1)$$

which corresponds to Hamiltonian dynamics. We then record the draw $\theta^{(1)} \leftarrow \theta(\tau)$, reset $\theta(0) \leftarrow \theta(\tau)$, and draw a new $p(0) \sim \mathcal{N}(0, M)$. Repeating the process, we obtain a sequence of draws $\theta^{(0)}, \theta^{(1)}, \theta^{(2)}, \ldots$.

It is not generally possible to simulate the trajectory in Eq. (1) exactly, so numerical integration is used. For a given starting point $(\theta, p)$, integration step size $\epsilon > 0$, and number of steps $s$, the leapfrog integrator $L_\epsilon$ (Neal, 2011) is applied $s$ times, where $(\theta', p') = L_\epsilon(\theta, p)$ is defined by $p'_{1/2} = p - \frac{\epsilon}{2}\nabla U(\theta)$, followed by $\theta' = \theta + \epsilon M^{-1} \cdot p'_{1/2}$,, and finally, $p' = p'_{1/2} - \frac{\epsilon}{2}\nabla U(\theta')$. Applying $L^s_\epsilon(\theta, p)$ ($L_\epsilon$ applied $s$ times to $(\theta, p)$) we obtain a proposed point $(\widetilde{\theta}, \widetilde{p})$ that approximates the idealized HMC point $(\theta(\tau), p(\tau))$ at time $\tau = \epsilon s$. A Metropolis–Hastings (MH) accept-reject step is then introduced in order to leave $\bar{\pi}$ invariant; the proposal $(\widetilde{\theta}, \widetilde{p})$ is accepted with the standard Metropolis probability denoted $\alpha((\theta, p), (\widetilde{\theta}, \widetilde{p}))$ and otherwise we remain at $(\theta, p)$.

### 2.3 Generalized linear models

Suppose we are given $n$ observed pairs of covariates $x_i \in \mathbb{R}^d$ and responses $y_i \in \mathcal{Y} \subset \mathbb{R}$, $i \in \{1, 2, \ldots, n\}$. The independent random responses $Y_i$ are assumed to have a conditional density $f_{Y|X}$ parameterized in terms of the mean of the distribution; in a generalized linear model (GLM), the mean of the distribution of response $Y_i$ is assumed to depend only on the *linear predictor*

$x_i^\top \theta$, where $\theta \in \mathbb{R}^d$. We fix some *inverse link function*, $\mu$, and model the mean $\mathbb{E}[Y_i] = \mu(x_i^\top \theta) \equiv \mu_i$. The full likelihood is then $\mathcal{L}(\theta) = \prod_{i=1}^n f_{Y|X}(y_i \mid \mu_i)$, and the log-likelihood corresponding to data point $(x_i, y_i)$ is $\ell_i = \log f_{Y|X}(y_i \mid \mu_i)$. In the Bayesian framework, a prior with density $\pi_0$ is specified on the regression parameters, such that one obtains a posterior distribution with density $\pi(\theta) \propto \pi_0(\theta) \cdot \mathcal{L}(\theta)$.

### 2.4 Condition numbers and preconditioning

The *condition number* of a given positive definite matrix $M$ is defined as $\kappa(M) := \lambda_{\max}(M)/\lambda_{\min}(M)$ where $\lambda_{\max}(M)$ and $\lambda_{\min}(M)$ are the largest and smallest eigenvalues of $M$, respectively. We can generalize this definition to accommodate strongly log-concave distributions. For a strongly log-concave distribution $\pi$, the condition number of $\pi$ can be defined as:

$$\kappa(\pi) := \sup_{\theta \in \mathbb{R}^d} \left\| \nabla^2 U(\theta) \right\| \sup_{\theta \in \mathbb{R}^d} \left\| \nabla^2 U(\theta)^{-1} \right\| .$$

In particular, when $\pi$ is Gaussian $\mathcal{N}(\mu, \Sigma)$, we have $\kappa(\pi) = \kappa(\Sigma) = \lambda_{\max}(\Sigma)/\lambda_{\min}(\Sigma)$. Moving forward, we refer to $\kappa(\pi)$ as the *raw condition number*; we simply denote this by $\kappa$ when there is no ambiguity.

It is common to transform the target distribution $\pi$ with a preconditioning matrix $A$ in order to try to reduce the condition number. That is, given $\theta \sim \pi$ and a full rank $d \times d$ matrix $A$, we define $\pi_A$ such that $\theta_A = A\theta \sim \pi_A$. In the context of HMC, preconditioning with $A$ is equivalent to setting the mass matrix to $A^\top A$. One common approach to preconditioning is to set $A = \operatorname{diag}(\operatorname{Var}_\pi[\theta_j]^{-1/2})$. In this case, we refer to $\kappa(\pi_A)$ as the *correlation condition number*, denoted as $\kappa_{\text{cor}}(\pi)$, since the covariance and correlation matrix of $\pi_A$ are the same. It is often implicitly assumed that $\kappa_{\text{cor}} \le \kappa$, but this is not always true (Hird and Livingstone, 2023), as we also see in our experiments.

We finally introduce the notion of the *residual condition number*, $\kappa_{\text{r}}(\pi)$, which is defined as $\kappa_{\text{r}}(\pi) := \inf_{A \in \mathcal{D}} \kappa(\pi_A)$, where $\mathcal{D}$ is the set of full rank $d \times d$ diagonal matrices. By definition, $\kappa_{\text{r}} \le \kappa$. We will demonstrate in Section 4 that the convergence rate for Gibbs sampling depends on $\kappa_{\text{r}}$, as opposed to $\kappa$.

## 3 COMPUTE GRAPH GIBBS

In this section, we present a fast algorithm for Gibbs sampling exploiting the structure of the compute graph. When this algorithm is applied to Gibbs sampling of GLMs, it reduces the computation time for a full scan of $d$ updates from $O(d^2 n)$ to $O(dn)$. Special cases of this algorithm have been applied to Gibbs sampling of specific models (see, e.g., Mahani and Sharabiani (2015), Equations 14–16), but not to probabilistic programming languages as far as we are aware. Our compute graph formulation makes it possible to automate and generalize the use of the algorithm presented in this section.

To illustrate our technique, we turn to logistic regression with parameters $\theta = (\theta_1, \ldots, \theta_d)$ and data points $\{(x_i, y_i)\}_{i=1}^n$. The likelihood for this model is given by

$$\mathcal{L}(\theta) = \prod_{i=1}^n p_i(\theta)^{y_i} (1 - p_i(\theta))^{1-y_i},$$
$$p_i(\theta) = \frac{1}{1 + \exp\left(-x_i^\top \theta\right)}.$$

We now introduce a flavour of *compute graph* suitable for our purpose, namely a type of directed graph associated with a function $f(\theta_1, \ldots, \theta_d)$—see Fig. 1 for an example in the GLM context. First, for each variable $\theta_j$, we assume there is a corresponding *input* node in the graph such that no edge points into it. We call the graph vertices that are not input nodes the *compute nodes*. For each compute node $n$, we assume there is an associated operation, i.e., a function that takes as input the values given by the nodes $n'$ such that $(n' \to n)$ is an edge in the graph. We assume the compute graph has a single *sink node*, i.e., a node which has no outgoing edge. We also posit that the composition of the operations along the compute graph from inputs to sink yields a function that is identical to $f(\theta_1, \ldots, \theta_d)$.

In Fig. 1, we see that the part of the compute graph for calculating the likelihood associated with data point $x_i$ revolves around computing the term $x_i^\top \theta$. After a single-coordinate update of $\theta$, as would be done with the Gibbs sampler, many of the compute graph nodes hold the same value before and after the update (in Fig. 1, those that change are highlighted in red). We explain next how these invariant values allow us to attain an $O(d)$ speedup of the Gibbs sampler for GLMs.

We now introduce our approach for computing the likelihood of GLMs, which we call compute graph Gibbs (CGGibbs). This approach is based on the following idea: cache the scalar values of the linear predictors $x_i^\top \theta$ for each data point $i$ in a vector of length $n$, `cache`. This requires only $O(n)$ memory (this is negligible since the design matrix already requires storing $d \cdot n$ entries), but yields an $O(d)$ computational speedup. This approach is possible because the Gibbs sampler considers coordinate-wise updates of the form $\theta = (\theta_1, \ldots, \theta_j, \ldots, \theta_d) \mapsto \theta' = (\theta_1, \ldots, \theta_j', \ldots, \theta_d)$, and consequently likelihood updates have a special structure. For such updates, we have the following conve-

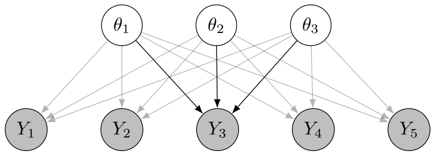


Figure 2: The *directed probabilistic graphical model* for the same regression problem as Fig. 1 with five data points $(x_i, Y_i)$ and three regression parameters $\theta_j$. Only edges into $Y_3$ are emphasized and the covariates $x_i$ are assumed fixed. Observed variables are shaded in gray. Each time a $\theta_j$ is changed, we have to recompute the likelihood from scratch for this representation as opposed to updating the cache at the '+' node found in the *compute graph* representation in Fig. 1.

nient decomposition

$$x_i^\top \theta' = \underbrace{\sum_{k=1}^{d} \theta_k x_{ik}}_{\texttt{cache}[i]} - \theta_j x_{ij} + \theta'_j x_{ij}.$$

This new linear predictor for a given data point can be computed in $O(1)$ time with caching, compared to $O(d)$ time without caching. From here, one evaluation of the full likelihood comes at an $O(n)$ cost, and a full sweep over $d$ coordinates is thus an $O(dn)$ cost, instead of the usual $O(d^2 n)$ that would be incurred without caching. The resulting speedup in terms of the number of parameters, $d$, allows us to apply this new approach to Bayesian inference with Gibbs sampling to high-dimensional regression problems, including ones where $d \gg n$.

Although this simple optimization is obvious from the *compute graph*, presented in Fig. 1, it is not obvious from the *directed graphical model* corresponding to the given regression problem in Fig. 2. Probabilistic programming languages (PPLs) incorporating Gibbs samplers often perform optimizations in evaluating ratios of likelihoods for Markov kernel proposals with respect to graphical models, such as the one in Fig. 2 (Lunn et al., 2009). However, these graphs only reveal the dependence structure of random variables in the model and decompositions of the likelihood, but they do not yield insight into fine-grained optimization of computations such as in Fig. 1. In our case, this "fine-grained" optimization yields a substantial $O(d)$ improvement in computation time with the introduction of simple caching techniques.

## 4 THEORY

Since GLMs come in many flavours of priors and likelihoods, we would ideally perform theoretical comparison of Gibbs and HMC under broad assumptions such as log-concavity. Unfortunately, current theoretical results for the log-concave setting are not mature enough—we seek to avoid the fallacy of comparing algorithms via loose complexity upper bounds. Specifically, current results for HMC and Gibbs on log-concave targets fall short on various aspects, such as failing to take into account both condition number and dimension simultaneously (Wang, 2017, 2019; Wang and Wu, 2014; Wang and Yin, 2020) or focusing on fixed integration time HMC instead of the non-constant integration time variants used in practice (Chen and Vempala, 2022). In fact, the scaling of randomized or dynamic HMC as a function of the condition number for log-concave targets is still an open problem as of the time of writing, with different authors positing conflicting conjectures (Lee et al., 2020; Apers et al., 2022).

Instead, we focus on normal models in this section. On one hand, Bayesian GLM posterior distributions are not exactly normal, but on the other hand, based on Bernstein-von Mises theorems (van der Vaart, 1998), many are well approximated by normal distributions. Other Bayesian GLM posterior distributions are not approximately normal, so we complement the results in this section with experiments on Bayesian GLMs with highly non-normal posterior distributions (Appendix A.8).

### 4.1 Convergence rates and scaling

**Gibbs convergence rate** Building on Roberts and Sahu (1997), we first establish the convergence rate of the idealized DUGS algorithm on Gaussian distributions with respect to various divergences: 1-Wasserstein, 2-Wasserstein distance, and the Pearson-$\chi^2$ divergence, denoted $\mathrm{TV}, \mathrm{W}_1, \mathrm{W}_2$ and $\chi^2$, respectively (see Appendix C for background on these divergences and how they relate to the empirical results in Section 5).

**Theorem 4.1.** *Let $\pi = \mathcal{N}(\mu, \Sigma)$ with precision matrix $\Sigma^{-1}$ having only non-positive off-diagonal elements. For any initial point $x \in \mathbb{R}^d$ and any $\mathrm{D} \in \{\mathrm{TV}, \mathrm{W}_1, \mathrm{W}_2, \chi^2\}$, we have*

$$\mathrm{D}(K^t_{DUGS}(x, \cdot), \pi) = O\left(\exp\left(-\frac{t}{\kappa(\Sigma)}\right)\right), \quad t \to \infty.$$

Theorem 4.1 shows that the contraction rate of DUGS is independent of $d$ for all four divergences. Concurrently to our work, Ascolani et al. (2024a) has proved a Kullback-Leibler divergence bound with the same convergence rate as in Theorem 4.1 for strongly log-concave and log-smooth targets and random sweep Gibbs samplers. Currently available bounds for randomized HMC do not appear as tight for strongly log-concave and log-smooth targets (see discussion on HMC convergence rate in the next paragraphs). Therefore, the comparison of theoretical results for Gibbs and HMC in the Gaussian case is still informative.

**Ideal HMC convergence rate** Most scaling results for *Metropolized HMC* are presented in terms of *mixing times* (which we review in the next section), and not on *convergence rates.* See Appendix C.2 for more discussion on convergence rates versus mixing times. There are, however, several studies on the convergence rates of *idealized HMC.* For example, Chen and Vempala (2022) provided a tight convergence rate bound of $O\left(1-\frac{1}{16\kappa(\pi)}\right)^t$ for idealized HMC with constant integration time on strongly log-concave and log-smooth targets $\pi$. However, Wang and Wibisono (2023) achieved a $\mathrm{W}_2$ contraction rate with improved $\kappa$ scaling—$O\left(1-\Theta\left(\frac{1}{\sqrt{\kappa}}\right)\right)^t$—for idealized HMC on Gaussian targets using a time-varying integration time called *Chebyshev integration time.* Jiang (2023) obtained the same accelerated rate on Gaussian targets using a random integration time combined with partial momentum refreshment. Whether these $O\left(1-\Theta\left(\frac{1}{\sqrt{\kappa}}\right)\right)^t$ rates of idealized HMC can be generalized to non-Gaussian targets remains an open question. Furthermore, although the rates in Wang and Wibisono (2023); Jiang (2023) are dimension-independent, idealized HMC is not implementable in practice since it requires exact ODE simulation.

**Metropolized HMC mixing time** The current state-of-the-art mixing rate for Metropolized HMC on general log-concave and log-smooth targets is $O(\kappa d^{1/4}\log(1/\epsilon))$ gradient queries for an $\epsilon$-level error in total variation distance (Chen and Gatmiry, 2023). The $O(d^{1/4})$ scaling (without known dependence on $\kappa$) was originally established in Beskos et al. (2013) for separable log-concave, log-smooth targets.

In another development, Chen et al. (2020) proved a mixing time of $O(\kappa d^{11/12})$ for general log-concave and log-smooth targets. Additionally, Lee et al. (2020) showed that HMC with a single leapfrog step per iteration has a mixing time of $O(\kappa d)$, and they conjectured that the $O(\kappa)$ dependence might be tight in this setting.

This dependence on the condition number has been improved in the case of Gaussian targets. Specifically, Apers et al. (2022) achieved a mixing time of $O(\kappa^{1/2}d^{1/4})$ by employing randomized integration steps at each iteration. This result mirrors the improved condition number scaling in the idealized HMC rates, and the authors of this study conjecture that this enhanced $\kappa$ scaling could potentially generalize to broader classes of log-concave and log-smooth targets, although this is yet to be studied.

## 4.2 Influence of diagonal preconditioning

The scaling results presented in Section 4.1 suggest that Gibbs sampling has better dimension scaling than Metropolized HMC with non-constant trajectory lengths, at least in the Gaussian case. In terms of condition number scaling the situation is more nuanced, and a full characterization requires a careful analysis of diagonal preconditioning, which we now discuss.

A first observation is that focusing solely on $\kappa(\pi)$, the condition number of the untransformed target, might not adequately represent the practical performance of the two samplers. While the observation described in the previous sentence is true for both samplers, its underlying cause is quite distinct for Gibbs and HMC, and so is the appropriate notion of condition number in each case.

For HMC, the standard practice is to fit a diagonal preconditioning matrix via adaptive MCMC; restricting to diagonal matrices ensures that the compute cost per leapfrog step stays linear in $d$. For instance, NUTS as implemented in Stan by default adopts diagonal preconditioning using estimated marginal standard deviations. Hence, for HMC run for enough iterations, the scaling in condition number will depend on the correlation condition number $\kappa_{\mathrm{cor}}$ introduced in Section 2.4, but with the caveat that marginal estimates need to be learned. Therefore, for early iterations the dependence will be on $\kappa$ rather than $\kappa_{\mathrm{cor}}$. On the other hand, thanks to the suppression of random walks brought by HMC with long trajectories, it is possible to achieve $\kappa^{1/2}$ scaling as discussed in Section 4.1, provided that HMC's key tuning parameters, the integrator step size and trajectory length, are well-tuned.

The situation for Gibbs brings a mix of good news and bad news, as we formalize in Proposition 4.2. On the negative side, the dependence grows linearly in the condition number instead of as a square root. On the positive side, because Gibbs is invariant to axis-aligned stretching, it is as if Gibbs automatically uses the optimal diagonal preconditioner, without having to explicitly learn it.

**Proposition 4.2.** *Under the same conditions as Theorem 4.1, for any initial point* $x\in\mathbb{R}^d$ *and any* $\mathrm{D}\in\{\mathrm{TV},\mathrm{W}_1,\mathrm{W}_2,\chi^2\}$,

$$\mathrm{D}(K^t_{DUGS}(x,\cdot),\pi)=O\left(\exp\left(-\frac{t}{\kappa_{\mathrm{r}}(\Sigma)}\right)\right),\quad t\to\infty.$$

In contrast, *HMC may even experience negative effects from diagonal preconditioning* (or equivalently, mass matrix adaptation). This was pointed out previously by Hird and Livingstone (2023, Sec 3.5.3), who provide an instance where diagonal preconditioning using the target standard deviation results in a worse condition number, even in the Gaussian case.

Even when such preconditioning theoretically offers benefits, current mass matrix adaptation methods rely on moment-based estimates, which can take a considerable amount of time to become accurate. Therefore, one might not achieve the idealized $\kappa$ rate in practice with HMC due to suboptimal mass matrix adaptation (see Appendix A.7).

Other works have also considered condition numbers resulting from different preconditioning strategies. Ascolani et al. (2024a) used the condition number after each dimension has been scaled by the square root of their respective smoothness constant. Hird and Livingstone (2023) showed how the condition number is affected by various linear preconditioning strategies such as scaling by the square root of the Fisher information matrix, the covariance matrix or using the R matrix from the QR decomposition, etc.

## 5 EXPERIMENTS

In this section, we conduct experiments showing the performance gain of CGGibbs over other popular Gibbs sampler implementations as well as comparing the efficiency of CGGibbs and NUTS on real datasets. The source code for these experiments can be found at `https://github.com/UBC-Stat-ML/gibbs-vs-hmc-mev`. All experiments were conducted on the ARC Sockeye computer cluster at the University of British Columbia. Additional experiments, details of the experimental setup and instructions for reproducibility are available in Appendix A.

### 5.1 Compute graph Gibbs is faster than prevailing Gibbs implementations

We first study the running time of various implementations of within-Gibbs samplers: our CGGibbs sampler, as well as the popular MultiBUGS v2.0 (Lunn et al., 2009), JAGS v4.3.2 (Plummer et al., 2003) and Gen.jl v0.4.7 (Cusumano-Towner et al., 2019) software packages. The goal is to confirm that previous implementations of within-Gibbs samplers scale in $O(d^2)$ per sweep, versus $O(d)$ for our method. For this experiment, we consider a sequence of synthetic logistic regression datasets with increasing dimension $d \in \{2^1, \dots, 2^{12}\}$. Each parameter has a Gaussian prior with standard deviation 10. Since in this first experiment we restrict ourselves to Gibbs samplers, we use the time taken to run 1000 sweeps as a representation of the computational complexity. The results, presented in Fig. 3, show that CGGibbs does indeed achieve an $O(d)$ scaling, while other currently available and commonly used

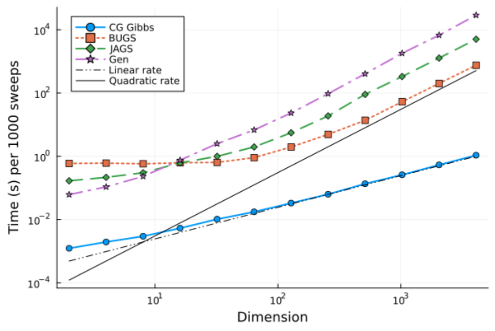


Figure 3: Wall-clock time (in seconds) taken to perform 1000 sweeps versus dimension for various Gibbs sampler implementations on synthetic logistic regression datasets of increasing dimensionality (log-log scale, lower is better).

Gibbs samplers have an undesirable $O(d^2)$ scaling.

### 5.2 Empirical scaling of compute graph Gibbs and NUTS

Next, we compare the wall-clock time per ESS as a function of the number of covariates for the Gibbs and Stan NUTS (2.35.0) (Carpenter et al., 2017) samplers. We exclude adaptation time from timings and focus on the stationary regime for simplicity. In the main text, we summarize the ESS performance by taking the median across all test functions $\theta \mapsto \theta_i$ (first moment of each marginal) and $\theta \mapsto \theta_i^2$ (squared marginal moments), see also Appendix A for results where ESS is summarized with the minimum ESS across test functions. We show results on the colon cancer dataset (Alon et al., 1999) in the main text, see Appendix A.4 for similar experiments on other datasets. For each dataset, we create a sequence of logistic regression problems with an increasing number of predictors and obtain samples from the posterior of each of these problems. Specifically, we shuffle uniformly at random the order of the covariates and choose an increasing prefix of $d \in \{2^1, 2^2, \dots, 2^{10}, 2000\}$ features from the permuted dataset (the colon dataset has 2000 features) to be predictors for the response and use an isotropic normal prior with standard deviation 10 for the parameters. Note that by adding covariates we change not only the dimensionality but also the condition number of the target distribution. The results are shown in Fig. 4, where we see that the efficiency of CGGibbs scales favourably as a function of the number of covariates compared to Stan NUTS.

Upon closer inspection of Fig. 4, there appears to be a change in behaviour in the performance of CGGibbs when $d \approx n$ (the colon dataset has $n = 62$ observations). To investigate this, we perform the same

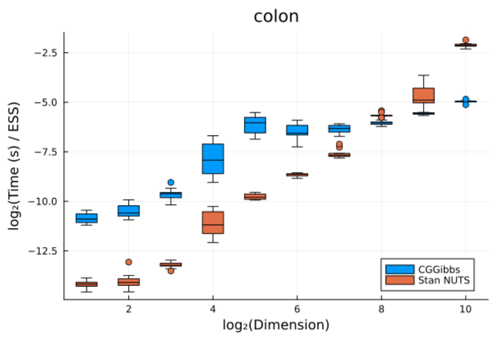


Figure 4: Wall-clock time (in seconds) per ESS for CGGibbs and Stan NUTS as a function of the number of subsampled predictors (log-log scale, lower is better) from a colon cancer gene expression dataset (Alon et al., 1999). Box plots summarize 10 replicates.

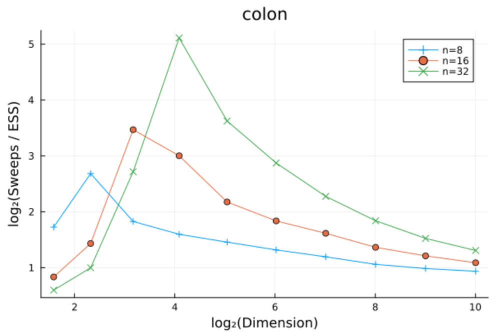


Figure 5: Dimensional scaling of the number of Gibbs sweeps to achieve one ESS for different subsampled size of data points (series colours) and predictors (abscissa).

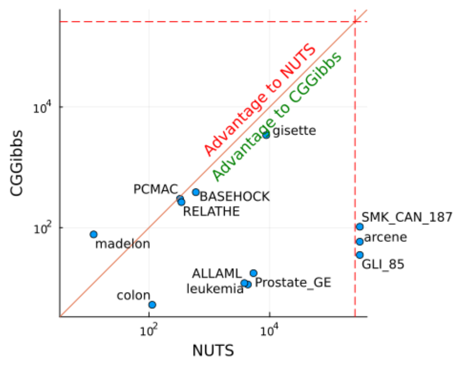


Figure 6: Median (across 30 replicates) wall-clock time (in seconds) per 100 ESS for CGGibbs and Stan NUTS on 12 logistic regression real datasets with a Gaussian prior. Each point is a dataset: its x-axis coordinate denotes the median time for NUTS, its y-axis coordinate, for CGGibbs. The region beyond the dashed lines indicates that the corresponding sampler has not reached a minimum of 100 ESS within three days.

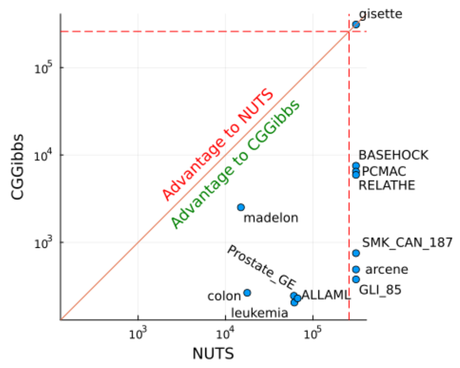


Figure 7: Similar plot to Fig. 6 but for the horseshoe prior.

experiment but with a varying number of observations with $n \in \{2^3, 2^4, 2^5\}$ and record the number of sweeps per median ESS. Fig. 5 confirms that there is again a similar transition when $d < n$ to $d > n$. This behaviour is partially explained with the stabilization (or decrease) of various notions of condition number when $d$ increases past $n$ (see Appendix A.5). Interestingly, the decrease in sweeps per ESS is similar to a result from Qin and Wang (2024, Theorem 2), where they developed a lower bound for the convergence rate of a random scan Gibbs sampler for submodels that depend on the full model. We further investigate this behaviour using controlled experiments with synthetic data in Appendix A.6.

### 5.3 Panel of datasets

We compare the time per ESS for CGGibbs and Stan NUTS on 12 binary classification datasets. These datasets include newsgroup datasets (Lang, 1995), gene expression datasets (Golub et al., 1999; Alon et al., 1999; Singh et al., 2002; Spira et al., 2007; Freije et al., 2004), a mass spectrometry data set (Guyon et al., 2004) a digit recognition data set (Guyon et al., 2004) and an artificial dataset (Guyon et al., 2004). We use a logistic regression model on each dataset with two types of priors: a Gaussian prior with a standard deviation of 10 for each of the parameters and a horseshoe prior (Carvalho et al., 2009). Inference for each combination of dataset and prior is repeated 30 times with different seeds. To avoid unreliable ESS estimates, we only report ESS estimates when the trace is sufficiently long to achieve an ESS of at least 100. More details on the data and model can be found in Appendix A.1.

The outcome of these experiments is summarized in Figs. 6 and 7. Overall, CGGibbs outperforms Stan NUTS in almost all datasets and is more than 300 times faster than Stan NUTS in time per ESS for the best case. Furthermore, Fig. 8 suggests a correlation between the $d/n$ ratio and the performance of CGGibbs relative to NUTS.

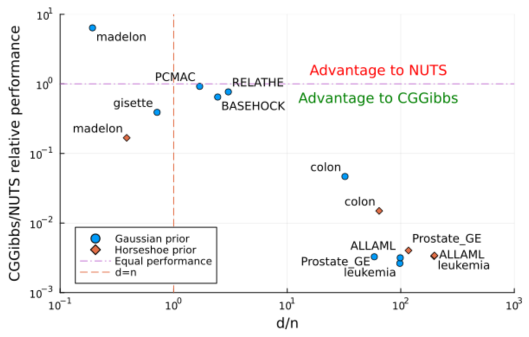


Figure 8: Ratio of median time (in seconds) per ESS between CGGibbs and Stan NUTS versus the $d/n$ ratio. Each point represents a combination of dataset and prior where both samplers reached 100 minimum ESS within 3 days. The dashed line indicates where $d = n$ and the dash dotted line indicates the threshold of equal performance between the two samplers.

## 6 DISCUSSION

Both our CGGibbs algorithm and reverse-mode automatic differentiation, (the key ingredient to automate the use of HMC in PPLs (Margossian, 2019)) are based on related but distinct notions of compute graphs. While we have focused on GLMs for concreteness, the efficient update of the compute graph described in Section 3 appears to apply more generally. Automatic processing of arbitrary compute graphs for efficient CGGibbs updates is potentially simpler than the machinery needed for reverse-mode automatic differentiation: in the former case, only certain reduce operations such as addition need special treatment, whereas in the latter case, all primitive operations need to be handled individually (i.e., each provided with an adjoint). Beyond GLMs, other examples of uses of CGGibbs include models with sufficient statistics and models of infinite dimensionality. Another interesting future direction is the analysis of algorithms combining the strengths of both Gibbs and HMC, such as optimal designs of block HMC-within-Gibbs algorithms.

### Acknowledgements

ABC and TC acknowledge the support of an NSERC Discovery Grant. NS acknowledges the support of a Vanier Canada Graduate Scholarship. We additionally acknowledge use of the ARC Sockeye computing platform from the University of British Columbia.

### References

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750.

Andrieu, C. (2016). On random- and systematic-scan samplers. *Biometrika*, 103(3):719–726.

Apers, S., Gribling, S., and Szilágyi, D. (2022). Hamiltonian Monte Carlo for efficient Gaussian sampling: Long and random steps. *arXiv:2209.12771*.

Ascolani, F., Lavenant, H., and Zanella, G. (2024a). Entropy contraction of the Gibbs sampler under log-concavity. *arXiv:2410.00858*.

Ascolani, F., Roberts, G. O., and Zanella, G. (2024b). Scalability of Metropolis-within-Gibbs schemes for high-dimensional Bayesian models. *arXiv:2403.09416*.

Beskos, A., Pillai, N. S., Roberts, G. O., Sanz-Serna, J. M., and Stuart, A. M. (2013). Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, 19(5A):1501–1534.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, 76:1–32.

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2009). Handling sparsity via the horseshoe. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 73–80. PMLR.

Chen, Y., Dwivedi, R., Wainwright, M. J., and Yu, B. (2020). Fast mixing of Metropolized Hamiltonian Monte Carlo: Benefits of multi-step gradients. *Journal of Machine Learning Research*, 21(92):1–72.

Chen, Y. and Gatmiry, K. (2023). When does Metropolized Hamiltonian Monte Carlo provably outperform Metropolis-adjusted Langevin algorithm? *arXiv:2304.04724*.

Chen, Z. and Vempala, S. S. (2022). Optimal convergence rate of Hamiltonian Monte Carlo for strongly logconcave distributions. *Theory of Computing*, 18:1–18.

Chib, S. and Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335.

Cusumano-Towner, M. F., Saad, F. A., Lew, A. K., and Mansinghka, V. K. (2019). Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 221–236. PLDI.

Flegal, J. M., Haran, M., and Jones, G. L. (2008). Markov Chain Monte Carlo: Can we trust the third significant figure? *Statistical Science*, 23(2):250–260.

Freije, W. A., Castro-Vargas, F. E., Fang, Z., Horvath, S., Cloughesy, T., Liau, L. M., Mischel, P. S., and Nelson, S. F. (2004). Gene expression profiling of gliomas strongly predicts survival. *Cancer Research*, 64(18):6503–6510.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. Chapman and Hall/CRC, 3rd edition edition.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., et al. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537.

Greenwood, P. E., McKeague, I. W., and Wefelmeyer, W. (1998). Information bounds for Gibbs samplers. *The Annals of Statistics*, 26(6):2128–2156.

Guyon, I., Gunn, S., Ben-Hur, A., and Dror, G. (2004). Result analysis of the NIPS 2003 feature selection challenge. *Advances in Neural Information Processing Systems*, 17.

He, B. D., De Sa, C. M., Mitliagkas, I., and Ré, C. (2016). Scan order in Gibbs sampling: Models in which it matters and bounds on how much. *Advances in Neural Information Processing Systems*, 29.

Hird, M. and Livingstone, S. (2023). Quantifying the effectiveness of linear preconditioning in Markov chain Monte Carlo. *arXiv:2312.04898*.

Hoffman, M. D. and Gelman, A. (2014). The No-U-Turn Sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(47):1593–1623.

Jiang, Q. (2023). On the dissipation of ideal Hamiltonian Monte Carlo sampler. *Stat*, 12(1).

Jordan, M. I. (2004). Graphical Models. *Statistical Science*, 19(1):140–155.

Lang, K. (1995). NewsWeeder: Learning to filter netnews. In *Machine Learning Proceedings*, pages 331–339. Elsevier.

Langmore, I., Dikovsky, M., Geraedts, S., Norgaard, P., and Von Behren, R. (2019). A condition number for Hamiltonian Monte Carlo. *arXiv:1905.09813*.

Lee, Y. T., Shen, R., and Tian, K. (2020). Logsmooth gradient concentration and tighter runtimes for Metropolized Hamiltonian Monte Carlo. In *Proceedings of the 33rd Conference on Learning Theory*, pages 2565–2597. PMLR.

Li, K. and Geng, Z. (2005). Convergence rate of Gibbs sampler and its application. *Science in China Series A: Mathematics*, 48(10):1430–1439.

Livingstone, S. and Zanella, G. (2022). The Barker Proposal: Combining Robustness and Efficiency in Gradient-Based MCMC. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(2):496–523.

Lunn, D., Spiegelhalter, D., Thomas, A., and Best, N. (2009). The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, 28(25):3049–3067.

Mahani, A. S. and Sharabiani, M. T. (2015). SIMD parallel MCMC sampling with applications for big-data Bayesian analytics. *Computational Statistics & Data Analysis*, 88:75–99.

Margossian, C. C. (2019). A review of automatic differentiation and its efficient implementation. *WIREs Data Mining and Knowledge Discovery*, 9(4):e1305.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer.

Neal, R. M. (2003). Slice sampling. *The Annals of Statistics*, 31(3):705–767.

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman and Hall/CRC.

Papaspiliopoulos, O., Roberts, G. O., and Zanella, G. (2020). Scalable inference for crossed random effects models. *Biometrika*, 107(1):25–40.

Plummer, M. et al. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, pages 1–10.

Qin, Q. and Jones, G. L. (2022). Convergence rates of two-component MCMC samplers. *Bernoulli*, 28(2):859–885.

Qin, Q. and Wang, G. (2024). Spectral telescope: Convergence rate bounds for random-scan Gibbs samplers based on a hierarchical structure. *The Annals of Applied Probability*, 34(1B):1319–1349.

Roberts, G. O. and Rosenthal, J. S. (2004). General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20 – 71.

Roberts, G. O. and Rosenthal, J. S. (2015). Surprising convergence properties of some simple Gibbs samplers under various scans. *International Journal of Statistics and Probability*, 5(1):51–60.

Roberts, G. O. and Sahu, S. K. (1997). Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler. *Journal of the Royal*

*Statistical Society Series B: Statistical Methodology*, 59(2):291–317.

Román, J. C., Hobert, J. P., and Presnell, B. (2014). On reparametrization and the Gibbs sampler. *Statistics & Probability Letters*, 91(C):110–116.

Singh, D., Febbo, P. G., Ross, K., Jackson, D. G., Manola, J., Ladd, C., Tamayo, P., Renshaw, A. A., D'Amico, A. V., Richie, J. P., et al. (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1(2):203–209.

Song, W. T. and Schmeiser, B. W. (1995). Optimal mean-squared-error batch sizes. *Management Science*, 41(1):110–123.

Spira, A., Beane, J. E., Shah, V., Steiling, K., Liu, G., Schembri, F., Gilman, S., Dumas, Y.-M., Calner, P., Sebastiani, P., et al. (2007). Airway epithelial gene expression in the diagnostic evaluation of smokers with suspect lung cancer. *Nature Medicine*, 13(3):361–366.

Surjanovic, N., Biron-Lattes, M., Tiede, P., Syed, S., Campbell, T., and Bouchard-Côté, A. (2023). Pigeons. jl: Distributed sampling from intractable distributions. *arXiv:2308.09769*.

Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398):528–540.

van der Vaart, A. W. (1998). *Asymptotic Statistics*. Cambridge University Press.

Wang, J.-K. and Wibisono, A. (2023). Accelerating Hamiltonian Monte Carlo via Chebyshev integration time. In *Proceedings of the 11th International Conference on Learning Representations*. ICLR.

Wang, N.-Y. (2017). Convergence rates of the random scan Gibbs sampler under the Dobrushin's uniqueness condition. *Electronic Communications in Probability*, 22(56):1–7.

Wang, N.-Y. (2019). Convergence rates of symmetric scan Gibbs sampler. *Frontiers of Mathematics in China*, 14(5):941–955.

Wang, N.-Y. and Wu, L. (2014). Convergence rate and concentration inequalities for Gibbs sampling in high dimension. *Bernoulli*, 20(4):1698–1716.

Wang, N.-Y. and Yin, G. (2020). Convergence rates of the blocked Gibbs sampler with random scan in the Wasserstein metric. *Stochastics*, 92(2):265–274.

Zanella, G. and Roberts, G. (2021). Multilevel linear models, Gibbs samplers and multigrid decompositions (with discussion). *Bayesian Analysis*, 16(4):1309–1391.

Štrumbelj, E., Bouchard-Côté, A., Corander, J., Gelman, A., Rue, H., Murray, L., Pesonen, H., Plummer, M., and Vehtari, A. (2024). Past, Present and Future of Software for Bayesian Inference. *Statistical Science*, 39(1):46–61.

## Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [**Yes**/No/Not Applicable]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [**Yes**/No/Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [**Yes**/No/Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [**Yes**/No/Not Applicable]
   (b) Complete proofs of all theoretical results. [**Yes**/No/Not Applicable]
   (c) Clear explanations of any assumptions. [**Yes**/No/Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [**Yes**/No/Not Applicable]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [**Yes**/No/Not Applicable]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [**Yes**/No/Not Applicable]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [**Yes**/No/Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [**Yes**/No/Not Applicable]
   (b) The license information of the assets, if applicable. [**Yes**/No/Not Applicable]

   (c) New assets either in the supplemental material or as a URL, if applicable. [**Yes**/No/Not Applicable]
   (d) Information about consent from data providers/curators. [Yes/No/**Not Applicable**]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/**Not Applicable**]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Yes/No/**Not Applicable**]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/**Not Applicable**]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/**Not Applicable**]

# Supplementary Materials

## A Details of experiments

In this section, we provide some additional experiments and details on the experiments found in the main text of the paper.

The license information for used assets is as follows:

- **Datasets**: We used the binary classification datasets from https://jundongl.github.io/scikit-feature/datasets.html, licensed under the GNU General Public License Version 2 (GPL-2.0).
- **multiBUGS v2.0:** GNU Lesser General Public License Version 3 (LGPL-3.0).
- **JAGS 4.3.2:** GNU General Public License Version 2.0 (GPL-2.0), MIT License.
- **Stan NUTS 2.35.0:** The 3-Clause BSD License (BSD 3-clause).
- **Julia package Pigeons:** GNU Affero General Public License Version 3 (AGPL-3.0).

### A.1 Details on Section 5.3

In this section, we give more details on the datasets as well as model specifics and data preprocessing procedures used in Section 5.

**Data details** We used the binary classification datasets from https://jundongl.github.io/scikit-feature/datasets.html, where a collection of classification datasets of varying sizes and sparsity is listed, along with their sources. The datasets we used vary in size from $n = 60$ to $n = 7000$ observations, as well as from $d = 500$ to $d = 22283$ predictors. Across the datasets, $d/n$ ranges from 0.1927 to 262.1647. These details are summarized in Table 1. Note that the sparsity is defined as the proportion of zero entries in the design matrix.

**Priors** The priors we use in our models are as follows:

**Gaussian prior** A Gaussian with mean 0 and standard deviation 10 is used for each parameter, i.e.

$$\theta_j \sim N(0, 10^2), \quad \forall\, j \in \{1, \dots, d\}.$$

**Horseshoe prior** A $t$-distribution prior is used for the intercept and zero mean Gaussian priors with standard deviation coming from a half-Cauchy are placed on the rest of the coefficients. That is,

$$\begin{aligned} \theta_1 &\sim t(3, 0, 1) \\ \theta_j &\sim N(0, \lambda_j^2 \tau^2), \quad \forall\, j \in \{2, \dots, d\} \\ \lambda_j &\sim C^+(0, 1), \quad \forall\, j \in \{2, \dots, d\} \\ \tau &\sim C^+(0, 1), \end{aligned}$$

where $t(\nu, \mu, \sigma)$ denotes a t-distribution with $\nu$ degrees of freedom, location $\mu$, scale $\sigma$ and $C^+(0, 1)$ denotes a half-Cauchy distribution. See Fig. 16 in Appendix A.8 for a visual representation of the complex geometry induced by this prior.

**Likelihood** We use a logistic regression likelihood, see Section 3.

**Algorithm 1** Compute graph Gibbs for GLMs ($T$ passes over all parameters)

```
Require: Initial regression parameters $\beta^{(0)}$, prior $\pi_0 = \otimes_{j=1}^{d} \pi_{0,j}$, observations $\{(x_i, y_i)\}_{i=1}^{n}$, inverse link function $\mu$, response distribution $f_{Y|X}$ (mean parametrization), # MCMC iterations $T$, conditional Markov kernel proposals $\{Q_{j|-j}\}_{j=1}^{d}$, conditional targets $\{\pi_{j|-j}\}_{j=1}^{d}$
 1: for $i$ in 1, 2, ..., $n$ do                                   ▷ One-time $O(dn)$ cache of linear predictors
 2:     cache[$i$] $\leftarrow \sum_{j=1}^{d} \beta_j x_{ij}$
 3: end for
 4: $\beta \leftarrow \beta^{(0)}$
 5: for $t$ in 1, 2, ..., $T$ do
 6:     for $j$ in 1, 2, ..., $d$ do
 7:         $\beta'_j \sim Q_{j|-j}(\cdot|\beta_{-j})$                  ▷ Metropolis-within-Gibbs proposal
 8:         $\ell \leftarrow 0$
 9:         for $i$ in 1, 2, ..., $n$ do                           ▷ Increment log-likelihood
10:             linear_predictor $\leftarrow$ cache[$i$] $- \beta_j x_{ij} + \beta'_j x_{ij}$
11:             $\ell \leftarrow \ell + \log(f_{Y|X}(y_i|\mu($linear_predictor$)))$
12:         end for
13:         $U \leftarrow \mathrm{Unif}(0, 1)$
14:         $\alpha \leftarrow$ MH_probability$(\ell, \pi_{j|-j}, Q(\cdot|\beta_{-j}), \pi_{0,j})$     ▷ Acceptance probability
15:         if $U \leq \alpha$ then                                 ▷ Accept proposal
16:             for $i$ in 1, 2, ..., $n$ do                       ▷ $O(n)$ update of cache
17:                 cache[$i$] $\leftarrow$ cache[$i$] $- \beta_j x_{ij} + \beta'_j x_{ij}$
18:             end for
19:             $\beta_j \leftarrow \beta'_j$                          ▷ Update parameter
20:         end if
21:     end for
22:     $\beta^{(t)} \leftarrow \beta$
23: end for
24: return $\{\beta^{(t)}\}_{t=0}^{T}$
```

Table 1: Datasets considered in our experiments.

| **Dataset** | **Sample size** | **Features** | **Sparsity** |
|---|---|---|---|
| ALLAML | 72 | 7129 | $3.3\times10^{-5}$ |
| BASEHOCK | 1993 | 4862 | 0.9861 |
| GLI_85 | 85 | 22283 | 0 |
| PCMAC | 1943 | 3289 | 0.9854 |
| Prostate_GE | 102 | 5966 | 0 |
| RELATHE | 1427 | 4322 | 0.9805 |
| SMK_CAN_187 | 187 | 19993 | 0 |
| arcene | 200 | 10000 | 0.4562 |
| colon | 62 | 2000 | 0.4158 |
| gisette | 7000 | 5000 | 0.87 |
| leukemia | 72 | 7070 | 0.4368 |
| madelon | 2600 | 500 | $7.6\times10^{-7}$ |

## A.2 CGGibbs algorithm

The CGGibbs algorithm when applied to GLMs is stated in Algorithm 1.

## A.3 Implementation details

All MCMC chains are run until a minimum ESS of 100 is reached. Here both CGGibbs and Stan NUTS use half of the total number of iterations as warmup and warmup samples are not used in the ESS calculations. The reasoning for why the ESS provides an adequate estimate of divergence scalings is given in Appendix C.3. We note that in our simulations, when subsampling is performed for each replicate, we change the shuffling of the data as well as the seed for the MCMC chains.

**CGGibbs** We use the slice sampler Neal (2003) from the Julia package `Pigeons` (Surjanovic et al., 2023) and set the number of passes through all variables per exploration step as 1 while the other hyperparameters are set to their default values.

**Stan NUTS** We use Stan NUTS (2.35.0) Carpenter et al. (2017) and set all hyperparameters other than warmup time to their defaults.

**Data preprocessing** We follow the standard practice of standardizing the design matrix of each dataset before supplying them to the samplers. Surprisingly, we observe that Stan NUTS benefits from sparse design matrices (sparsity greater than 0.85) even without sparse encoding in the model. Therefore, we opt rescale the non-zero entries of sparse datasets by dividing by their maximum absolute values for each column for both Stan NUTS and CGGibbs, making the comparison as fair as possible.

## A.4 More dimensional scaling results

In this section, we repeat the increasing column experiment previously done for the colon cancer dataset on other datasets. These are datasets from Section 5.3 where both CGGibbs and Stan NUTS achieve a minimum ESS of 100 within 3 days using a Gaussian prior. For each dataset, there are 10 replicates runs with a different shuffling of the features and a different random seed for the MCMC chain. The results are shown in Figs. 9 and 10.

In addition, we also show the number of sweeps (or iterations for NUTS) per median ESS in Figs. 11 and 12, where the shift from $d < n$ to $d > n$ is more evident. Observe that this shift does not always occur at the point where $d = n$ but usually in its vicinity.

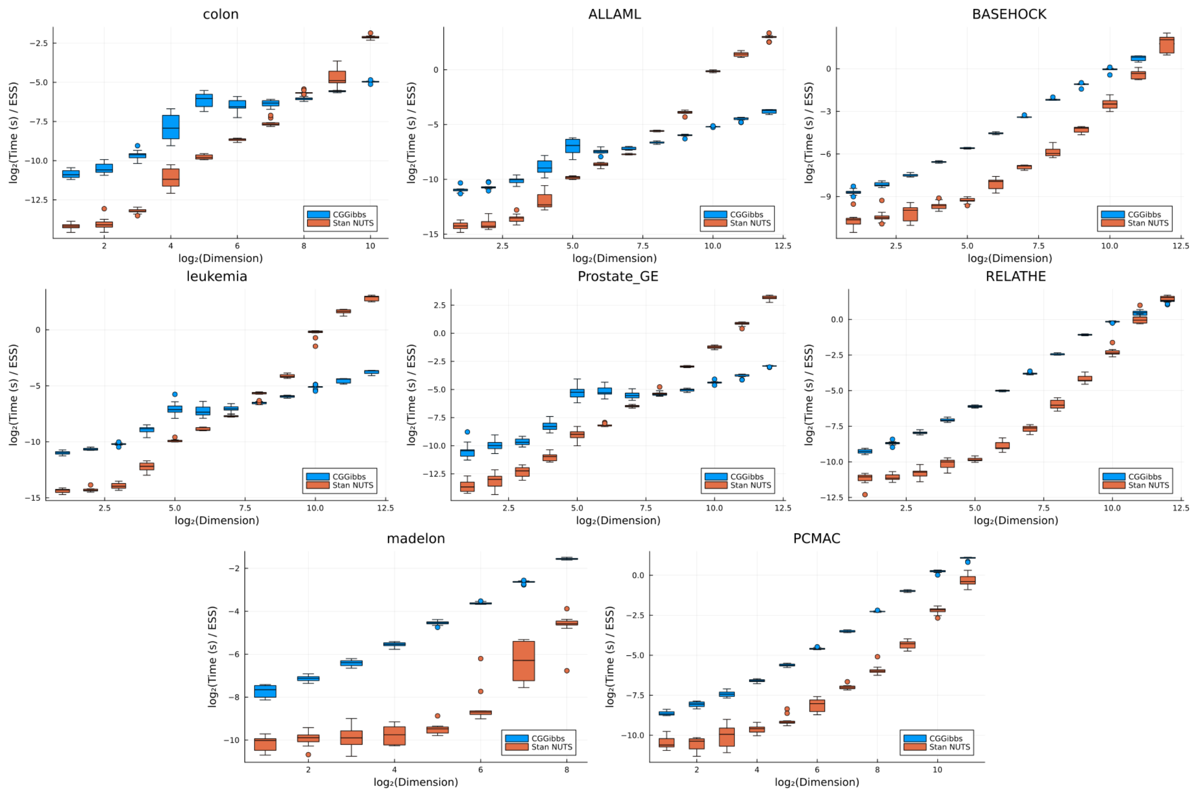


Figure 9: Time per median ESS for CGGibbs and Stan NUTS as a function of dimension for several other datasets. The box plots summarize the results over 10 replicates.

## A.5 Condition number scaling

In this section, we show several examples of the shift in sampling efficiency from $d < n$ to $d > n$ also occuring for various notions of condition number when the Gaussian prior is used. The results are shown in Fig. 13. Based on this figure, we conclude that the shift in sampling efficiency may not occur for sparse data such as the PCMAC newsgroup dataset.

## A.6 Impact of irrelevant parameters on the Gibbs mixing rate

To further examine the effect of varying the number of features on the Gibbs sampler, as seen in Section 5.2, we generate three synthetic datasets where only a prefix of the features influence the outcome. Specifically, each dataset has

$$\begin{aligned}\text{Design matrix:}\quad & x = (x_1, x_2, \ldots, x_n)^\top \in \mathbb{R}^{n\times d},\\ \text{True parameter:}\quad & \theta = (a, \theta_s, \mathbf{0}) \in \mathbb{R}^d,\\ \text{Logistic outcome:}\quad & y_i \mid x_i \sim \text{Bern}(\text{logistic}(a + \theta^\top x_i)), \quad i = 1, \ldots, n,\end{aligned}$$

where $d = 2^{15}, n = 20, x_i \in \mathbb{R}^d$ for all $i = 1, \ldots, n$, $a \in \mathbb{R}$ is the intercept and $\theta_s \in \mathbb{R}^{30}$ is the coefficient vector for the significant features. The design matrices for each scenario are as follows:

(1) Set $x_i \sim \mathcal{N}(0, I_d)$ for all $i = 1, \ldots, n$. In this setting, the features are uncorrelated.

(2) Set $x_i$ the same as scenario (1) for all $i \le 30$ and set $x_i = x_1$ for all $i > 30$. Here, extra features are perfectly correlated with the first (significant) feature.

(3) Set $x_i$ the same as scenario (1) for all $i \le 30$ and set $x_i = 0$ for all $i > 30$.

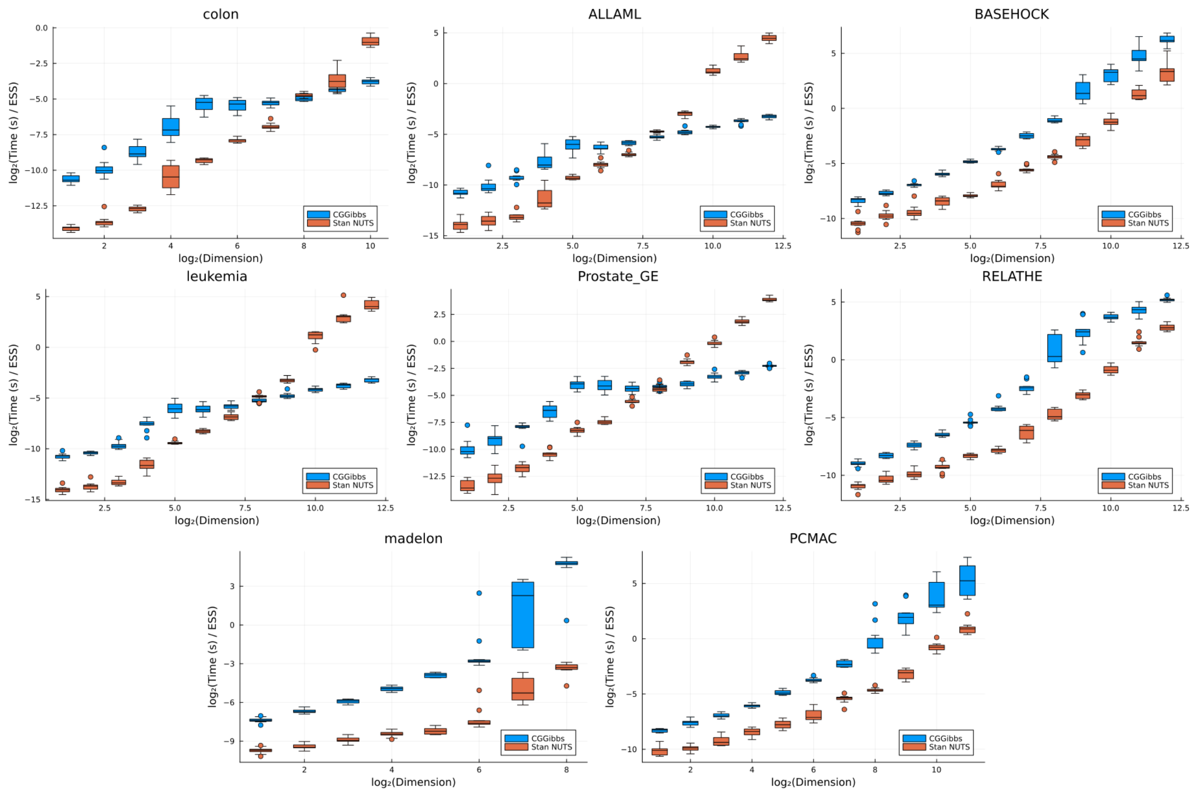


Figure 10: Time per min ESS for CGGibbs and Stan NUTS as a function of dimension for several other datasets. The box plots summarize the results over 10 replicates.

We use CGGibbs to sample from these synthetic datasets using a Gaussian prior. The results of these experiments are shown in Fig. 14. The results confirm the surprising findings of Section 5.2 that adding more features can help the Gibbs sampler even if the added features are not related to the data generation process—depending on how "well behaved" the extra features are. In the case of an overparameterized GLM, the irrelevant dimensions might be related to the auxiliary variables used in data augmentation samplers (Tanner and Wong, 1987). We suspect that the extra features let the parameters move more freely in a larger state space, thereby speeding up the overall mixing of the chain. Based on these results, it seems that the Gibbs sampler can potentially benefit from overparameterization.

In addition, zero elements can decrease the number of sweeps per median ESS while not helping decrease the number of sweeps per min ESS. This is because updates to components associated with zero elements are not affected by the likelihood, thereby making the chains of these components sample from the priors. As prior distributions are typically easy to sample from, the ESS of these components can increase, which increases the median ESS across marginals (but not the min ESS, which measures the most difficult marginal). We also observe this disparity between min and median ESS in sparse datasets in Appendix A.9

## A.7 Condition number adaptation with NUTS

In this section, we show an example of a problem where $\kappa_{\text{cor}} < \kappa$. For this problem, we apply a logistic regression model with a Gaussian prior on a subset of the prostate cancer dataset where $d = n = 16$ and record the progression of the condition number adaptation in Stan NUTS. The result is shown in Fig. 15.

## A.8 Horseshoe prior example

Fig. 16 shows a visual representation of the horseshoe prior for the case $\beta \in \mathbb{R}^2$. Notice that, while the univariate marginals would suggest a log-concave distribution, the bivariate densities indicate a much more complex geometry.

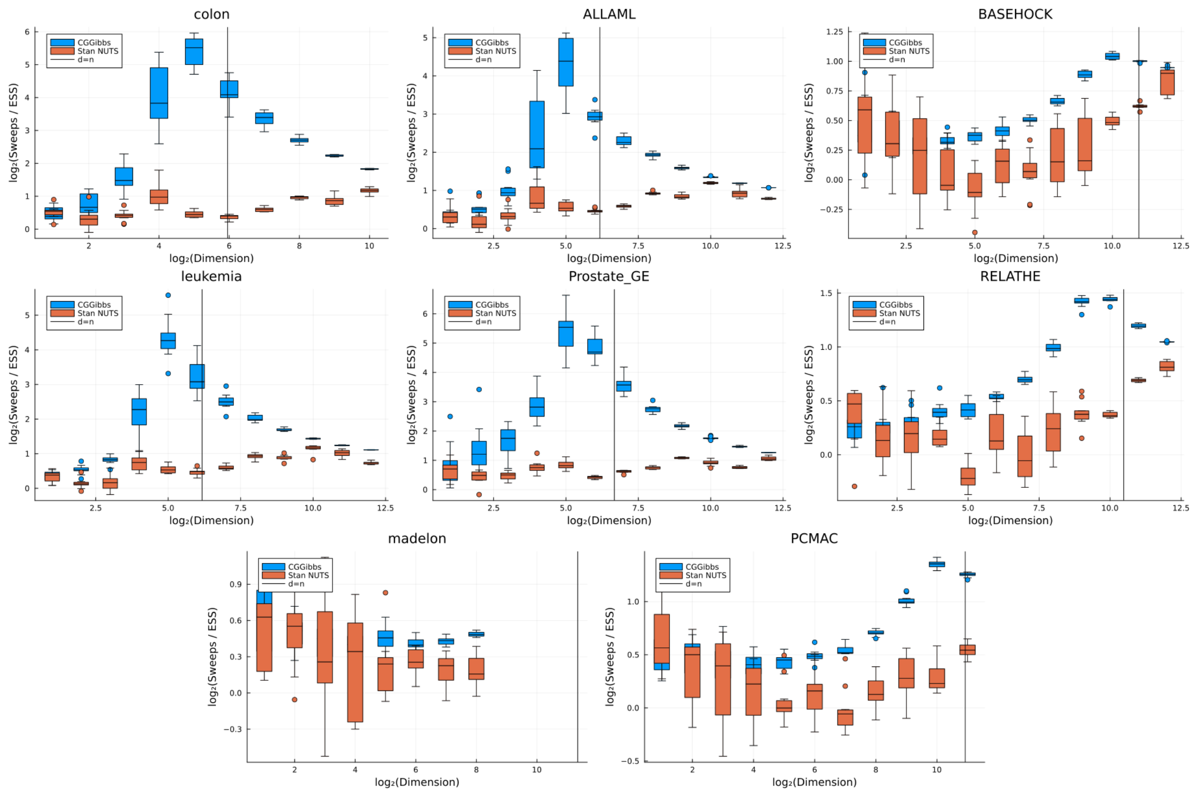


Figure 11: Sweeps per median ESS for CGGibbs and Stan NUTS as a function of dimension. The black line indicates the point at which $d = n$. The box plots summarize the results across 10 replicates.

### A.9 Panel of datasets: results for min ESS

The time per minimum ESS results for the panel of datasets considered in Section 5.3 are shown in Fig. 17. For non-sparse datasets, the performance for both samplers in terms of the minimum ESS is similar to the median ESS counterpart. For sparse data, however, Stan NUTS performs better in terms of time per minimum ESS. This behaviour is in line with the results from Appendix A.6 as a sparse data matrix can be interpreted as being similar to the third scenario in that section.

## B Proofs

In this section we prove Theorem 4.1 and Proposition 4.2.

### B.1 Proof of Theorem 4.1

*Proof of Theorem 4.1.* To establish the convergence rate of DUGS, we first define some relevant matrices. Let

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \dots & \Sigma_{1d} \\ \Sigma_{21} & \Sigma_{22} & \dots & \Sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{d1} & \Sigma_{d2} & \dots & \Sigma_{dd} \end{pmatrix}, \qquad \Sigma^{-1} := Q = \begin{pmatrix} Q_{11} & Q_{12} & \dots & Q_{1d} \\ Q_{21} & Q_{22} & \dots & Q_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{d1} & Q_{d2} & \dots & Q_{dd} \end{pmatrix}.$$

Define

$$A(\Sigma) = I - \text{diag}(Q_{11}^{-1}, Q_{22}^{-1}, \dots, Q_{dd}^{-1})Q,$$

and let $L(\Sigma)$ be a block lower triangular matrix such that the lower triangle blocks coincide with those of $A(\Sigma)$. Finally, set $U(\Sigma) = A(\Sigma) - L(\Sigma)$ and define

$$B(\Sigma) = (I - L(\Sigma))^{-1}U(\Sigma).$$

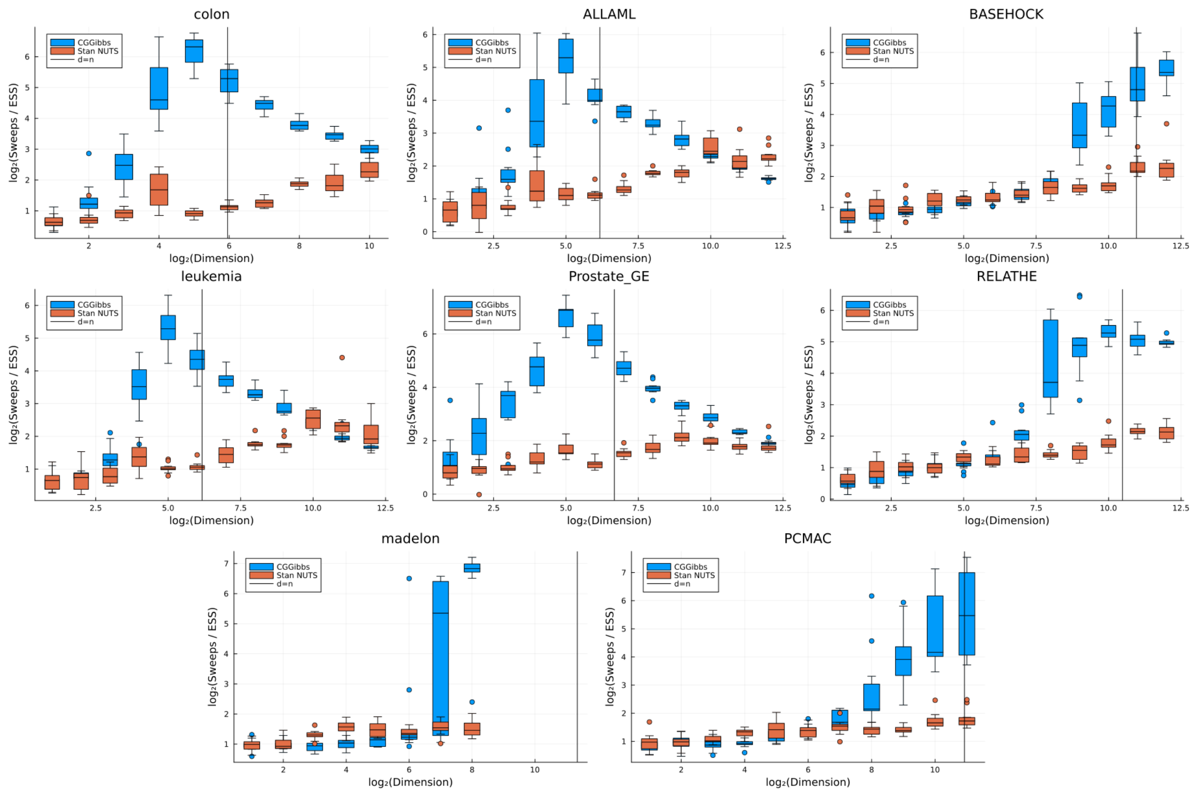


Figure 12: Number of sweeps per min ESS for CGGibbs and Stan NUTS as a function of dimension. The black line indicates the point at which $d = n$. The box plots summarize the results across 10 replicates.

We express the convergence rate for each metric in terms of the spectral radius of the matrix $B(\Sigma)$, denoted $\rho(B(\Sigma))$, which is the modulus of the eigenvalue of $B(\Sigma)$ with the largest modulus. From Lemma B.1, we have that

$$\rho(B(\Sigma)) \leq \exp\left(-\frac{1}{\kappa(\Sigma)}\right).$$

**TV bound.** First, we prove that the rate of convergence of DUGS in TV distance is given by $\rho(B(\Sigma))$. From Theorem 1 of Roberts and Sahu (1997), we have $\pi_t := K^t_{\text{DUGS}}(x, \cdot) = \mathcal{N}(\mu_t, \Sigma_t)$ for all $t \geq 0$, where

$$\mu_t = \mu + B(\Sigma)^t(\mu_0 - \mu), \qquad \Sigma_t = \Sigma + B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^\top)^t \tag{2}$$

with $\mathcal{N}(\mu_0, \Sigma_0)$ being the initial distribution. (I.e., here $\mu_0 = x$ and $\Sigma_0 = 0$.) By Pinsker's inequality, we can bound the total variation with the KL divergence, and the KL divergence between two Gaussians has a closed-form expression, so that

$$\begin{aligned} \text{TV}(\pi, \pi_t) &\leq \sqrt{\frac{1}{2}\text{KL}(\pi\|\pi_t)} \\ &= \frac{1}{\sqrt{2}}\sqrt{\text{tr}(\Sigma^{-1}\Sigma_t - I) + (\mu - \mu_t)^\top\Sigma^{-1}(\mu - \mu_t) - \log\det(\Sigma_t\Sigma^{-1})}. \end{aligned} \tag{3}$$

We analyze the convergence for each of the terms inside the square root. By Eq. (2),

$$\begin{aligned} \text{tr}(\Sigma^{-1}\Sigma_t - I) &= \text{tr}\left(\Sigma^{-1}(\Sigma + B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^t)^\top) - I\right) \\ &= \text{tr}(\Sigma^{-1}B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^t)^\top) \\ &= O(\rho(B(\Sigma))^{2t}), \end{aligned} \tag{4}$$

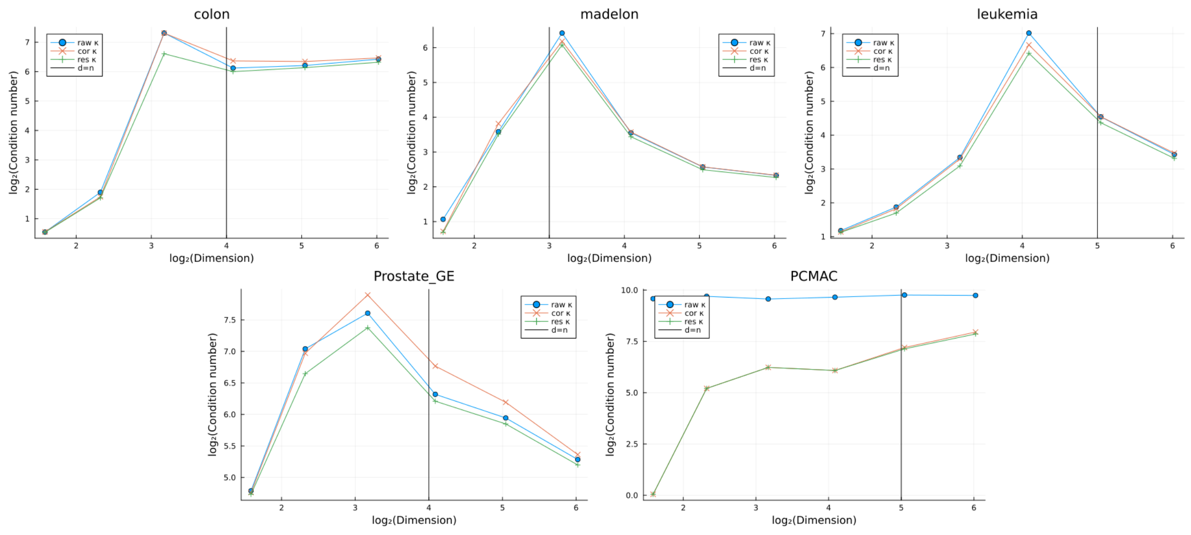


Figure 13: Dimensional scaling of various notions of condition number when the Gaussian prior is used for different datasets and sample sizes. The black line indicates the point at which $d = n$.

where the asymptotic rate is due to the fact that $B(\Sigma)^t$ converges to 0 element-wise at rate $\rho(B(\Sigma))$ (Roberts and Sahu, 1997, Lemma 4). Similarly, combining Eq. (2) and the convergence of $B(\Sigma)^t$, we have that

$$\|\mu - \mu_t\| = \|B^t(\mu - \mu_0)\| = O(\rho(B(\Sigma))^t),$$

yielding

$$(\mu - \mu_t)^\top \Sigma^{-1}(\mu - \mu_t) = O(\rho(B(\Sigma))^{2t}). \tag{5}$$

For the third term, we obtain that

$$\begin{aligned} |\log\det(\Sigma_t\Sigma^{-1})| &= |\log\det(I + B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^t)^\top\Sigma^{-1})| \\ &\le \operatorname{tr}(B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^t)^\top\Sigma^{-1}) \\ &= O(\rho(B(\Sigma))^{2t}). \end{aligned} \tag{6}$$

Finally, combining Eq. (3) and Eqs. (4) to (6) yields

$$\mathrm{TV}(\pi, \pi_t) = O(\rho(B(\Sigma))^t). \tag{7}$$

**Wasserstein bounds.** Next, we prove the Wasserstein bounds. For two Gaussian distributions $\pi_1 = \mathcal{N}(\mu_1, \Sigma_1)$ and $\pi_2 = \mathcal{N}(\mu_2, \Sigma_2)$, we have the following formula for their Wasserstein 2-distance:

$$\mathrm{W}_2^2(\pi_1, \pi_2) = \|\mu_1 - \mu_2\|_2^2 + \operatorname{tr}(\Sigma_1 + \Sigma_2 - 2(\Sigma_2^{1/2}\Sigma_1\Sigma_2^{1/2})^{1/2}), \tag{8}$$

where $\operatorname{tr}(M)$ is the trace of the matrix $M$ and $M^{1/2}$ is the principal square root of $M$. Applying (8) to $\pi_1 = \pi_t = \mathcal{N}(\mu_t, \Sigma_t)$ and $\pi_2 = \pi = \mathcal{N}(\mu, \Sigma)$ gives us

$$\begin{aligned} \mathrm{W}_2^2(\pi, \pi_t) &= \|\mu - \mu_t\|_2^2 + \operatorname{tr}(\Sigma + \Sigma_t - 2(\Sigma^{1/2}\Sigma_t\Sigma^{1/2})^{1/2}) \\ &= \|B(\Sigma)^t(\mu_0 - \mu)\|_2^2 + \operatorname{tr}(2\Sigma + B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^\top)^t) \\ &\qquad - 2\operatorname{tr}(\Sigma^{1/2}(\Sigma + B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^\top)^t)\Sigma^{1/2})^{1/2}) \\ &= \|B(\Sigma)^t(\mu_0 - \mu)\|_2^2 + 2\operatorname{tr}(\Sigma) + \operatorname{tr}(B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^\top)^t) \\ &\qquad - 2\operatorname{tr}((\Sigma^2 + (\Sigma^{1/2}B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^\top)^t\Sigma^{1/2})^{1/2}) \\ &= \|B(\Sigma)^t(\mu_0 - \mu)\|_2^2 + 2\operatorname{tr}(\Sigma) + \operatorname{tr}(M_t) - 2\operatorname{tr}((\Sigma^2 + (\Sigma^{1/2}M_t\Sigma^{1/2})^{1/2}), \end{aligned} \tag{9}$$

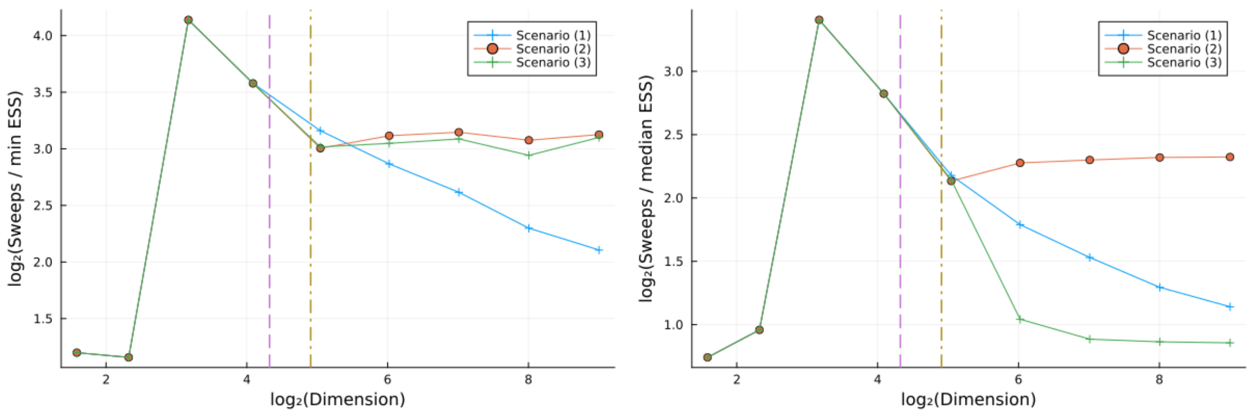


Figure 14: Effect of extra features on the number of Gibbs sweeps per ESS when the Gaussian prior is used in three different scenarios: (1) no multicollinearity in the design matrix, (2) multicollinearity present in the design matrix, and (3) irrelevant features are identically zero. The dashed line is where $d = n$ and the dash-dotted line is where the “irrelevant features” start to be added. **Left:** results in terms of sweeps per min ESS, **reft:** results in terms of sweeps per median ESS

where $M_t := B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^\top)^t$. We bound the last term in the above expression using the fact

$$\exists\ \epsilon \in (0,1): \ \operatorname{tr}\left(4\frac{1-\epsilon}{\epsilon^2}\Sigma - M_t\right) \geq 0 \quad \forall\ t \geq 0.$$

This is easy to confirm since the range of the function $g(\epsilon) = 4\frac{1-\epsilon}{\epsilon^2}$ in $(0,1)$ is $(0,\infty)$ and the matrix in the trace is symmetric. With this, we have

$$\begin{aligned}
&\operatorname{tr}\left(4\frac{1-\epsilon}{\epsilon^2}\Sigma - M_t\right) \geq 0.\\
\Leftrightarrow\ &\operatorname{tr}\left(4\frac{1-\epsilon}{\epsilon^2}\Sigma M_t - M_t^2\right) \geq 0.\\
\Leftrightarrow\ &\operatorname{tr}\left((1-\epsilon)\Sigma M_t - \frac{\epsilon^2}{4}M_t^2\right) \geq 0.\\
\Leftrightarrow\ &\operatorname{tr}\left(\Sigma M_t\right) \geq \operatorname{tr}\left(\epsilon\Sigma M_t + \frac{\epsilon^2}{4}M_t^2\right).\\
\Leftrightarrow\ &\operatorname{tr}\left(\Sigma^2 + \Sigma M_t\right) \geq \operatorname{tr}\left(\Sigma^2 + \epsilon\Sigma M_t + \frac{\epsilon^2}{4}M_t^2\right).\\
\Leftrightarrow\ &\operatorname{tr}\left(\Sigma^2 + \Sigma^{1/2}M_t\Sigma^{1/2}\right) \geq \operatorname{tr}\left(\left(\Sigma + \frac{\epsilon}{2}M_t\right)^2\right).
\end{aligned}$$

Finally, since the square root operation is monotone increasing, we have

$$\operatorname{tr}\left((\Sigma^2 + \Sigma^{1/2}M_t\Sigma^{1/2})^{1/2}\right) \geq \operatorname{tr}(\Sigma) + \frac{\epsilon}{2}\operatorname{tr}(M_t). \tag{10}$$

Plugging (10) into (9) gives us

$$\begin{aligned}
\mathrm{W}_2^2(\pi,\pi_t) &\leq \|B(\Sigma)^t(\mu_0 - \mu)\|_2^2 + 2\operatorname{tr}(\Sigma) + \operatorname{tr}(M_t) - 2\operatorname{tr}(\Sigma) - \epsilon\operatorname{tr}(M_t)\\
&= \|B(\Sigma)^t(\mu_0 - \mu)\|_2^2 + (1-\epsilon)\operatorname{tr}(M_t)\\
&= \|B(\Sigma)^t(\mu_0 - \mu)\|_2^2 + (1-\epsilon)\operatorname{tr}(B(\Sigma)^t(\Sigma_0 - \Sigma)(B(\Sigma)^\top)^t).
\end{aligned}$$

Combining this with Lemma 4 from Roberts and Sahu (1997), we have

$$\mathrm{W}_2^2(\pi,\pi_t) = O(\rho(B(\Sigma))^{2t})$$

which shows the convergence rate for Wasserstein 2 distance. By properties of the Wasserstein distance, it follows that $\mathrm{W}_1(\pi_t,\pi) \leq \mathrm{W}_2(\pi_t,\pi)$ and hence $\mathrm{W}_1(\pi_t,\pi) = O(\rho(B(\Sigma))^t)$.

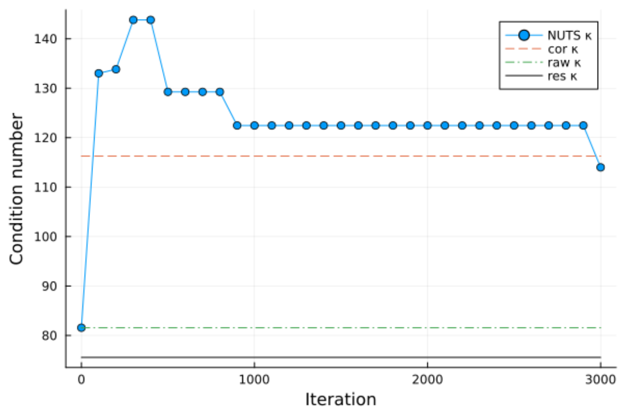


Figure 15: Condition number (lower is better) adaption used by NUTS for a logistic regression problem with a Gaussian prior. The condition number is plotted as a function of the number of MCMC iterations. The green dashed line is the initial condition number (no preconditioning), the red dashed line is the condition number after preconditioning with marginal standard deviations and the solid black line is the best linear preconditioner. The NUTS condition number (blue solid line) eventually converges to the red dashed line.

**Chi-squared bound.** For the Pearson-$\chi^2$ divergence, the result follows from direct application of Theorem 1 and Lemma 4 of Li and Geng (2005). Namely, we have that

$$\chi^2(\pi_t, \pi) = O(\rho(B(\Sigma))^t).$$

$\square$

**Lemma B.1.** *Under the conditions of Theorem 4.1,*

$$\rho(B(\Sigma)) \leq \exp\left(-\frac{1}{\kappa(\Sigma)}\right).$$

*Proof of Lemma B.1.* Our proof proceeds in two steps, using results from Roberts and Sahu (1997). We first establish a bound on the $L^2$ convergence rate of the random sweep Gibbs sampler (RSGS), $\rho_{L^2}(RSGS, \Sigma)$, in terms of $d$ and $\kappa(\Sigma)$. This rate is easier to study as it involves the matrix $A(\Sigma)$ instead of $B(\Sigma)$. We then conclude that $\rho(B(\Sigma)) \leq \rho_{L^2}(RSGS, \Sigma)$ from an existing result in Roberts and Sahu (1997).

For any symmetric positive definite matrix $\Sigma$, there exists a rotation matrix $R$ that makes $R^{-1}\Sigma R$ diagonal. Furthermore, the condition number $\kappa(\Sigma)$ of the Gaussian distribution with covariance $\Sigma$ is invariant to rotations. Therefore, we can consider $\mathcal{N}(\mu, \Sigma)$ with a fixed condition number, *diagonal* covariance matrix $\Sigma$ and its rotations without loss of generality. Now, for any matrix $M$, denote $\lambda(M)$ its set of eigenvalues and $D_M$ the inverse of the diagonal of $M$ if it has invertible diagonal elements. Let $\lambda_1 \geq \ldots \lambda_d > 0$ denote the eigenvalues of $\Sigma$ so that $\kappa(\Sigma) = \lambda_1/\lambda_d$. Rotating $\mathcal{N}(\mu, \Sigma)$ using a rotation matrix $R$ gives us $\mathcal{N}(R\mu, R\Sigma R^\top)$. The RSGS coordinate update matrix for $\mathcal{N}(R\mu, R\Sigma R^\top)$ is

$$A(R\Sigma R^\top) = I - D_{RQR^\top} RQR^\top.$$

To study the maximum eigenvalue of this matrix, we first note two matrix properties. First, for all symmetric positive definite matrices $M, N$, using the spectral norm, we have the inequality

$$\frac{1}{\min \lambda(MN)} = \|(MN)^{-1}\| \leq \|M^{-1}\|\|N^{-1}\| = \frac{1}{\min \lambda(M)} \frac{1}{\min \lambda(N)}.$$

Also, since $\sum_{j=1}^d r_{i,j}^2 = 1$ for the matrix $R$, we have

$$\sum_{j=1}^d \frac{r_{i,j}^2}{\lambda_j} \geq \sum_{j=1}^d \frac{r_{i,j}^2}{\lambda_1} = \frac{1}{\lambda_1}.$$

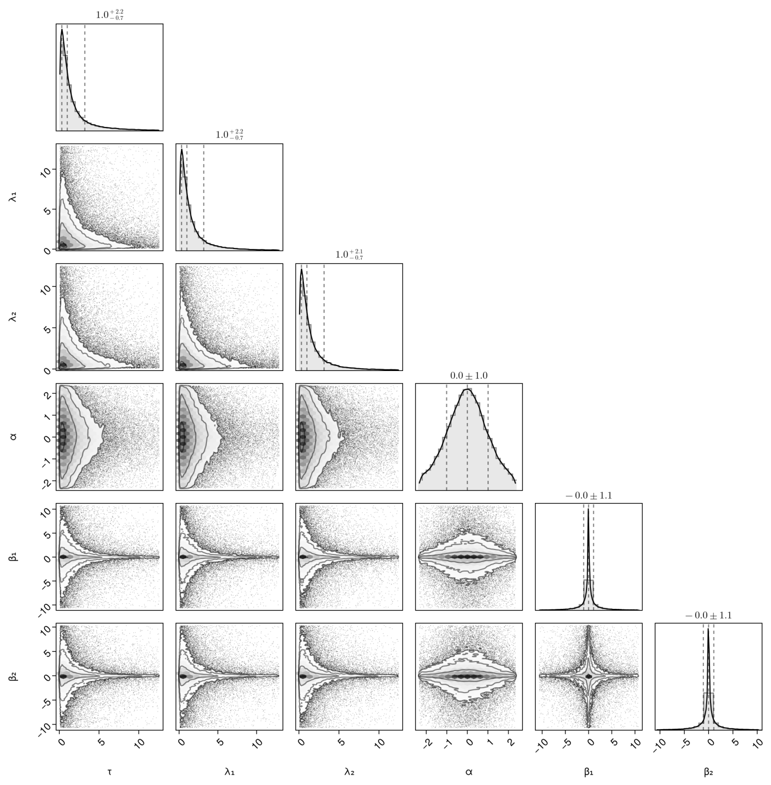


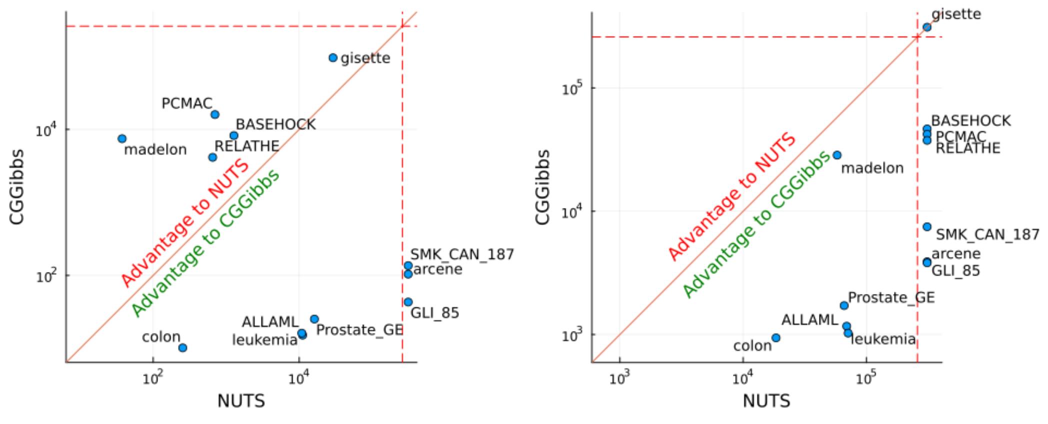
Figure 16: Univariate and bivariate marginals of the horseshoe prior for $\beta \in \mathbb{R}^2$, approximated by drawing 100,000 i.i.d. samples from the generative model.

Combining these results, we have

$$\begin{aligned}
\max \lambda(A(R\Sigma R^\top)) &= 1 - \min \lambda(D_{RQR^\top} RQR^\top) \\
&\leq 1 - \min \lambda(D_{RQR^\top}) \min \lambda(RQR^\top) \\
&= 1 - \min \lambda(D_{RQR^\top}) \min \lambda(Q) \\
&= 1 - \min \lambda(D_{RQR^\top}) \lambda_d \\
&= 1 - \lambda_d \min_i \sum_{j=1}^d \frac{r_{i,j}^2}{\lambda_j} \\
&= 1 - \frac{1}{\kappa(\Sigma)}.
\end{aligned}$$

Now, let $\mathrm{rot}_d$ be the set of $d$ by $d$ rotation matrices. By Theorem 2 in Roberts and Sahu (1997), we have

$$\begin{aligned}
\rho_{L^2}(RSGS, \Sigma) &\leq \sup_{R \in \mathrm{rot}_d} \rho_{L^2}(RSGS, R\Sigma R^\top) \qquad (11) \\
&= \sup_{R \in \mathrm{rot}_d} \left( \frac{1}{d}(d - 1 + \max \lambda(A(R\Sigma R^\top))) \right)^d \\
&\leq \left( \frac{1}{d} \left( d - 1 + 1 - \frac{1}{\kappa(\Sigma)} \right) \right)^d \\
&\leq \exp\left( -\frac{1}{\kappa(\Sigma)} \right).
\end{aligned}$$



(a) Gaussian prior (b) Horseshoe prior

Figure 17: Median (across 30 replicates) wall-clock time (in seconds) per 100 minimum ESS for CGGibbs and Stan NUTS on 12 logistic regression problems with real datasets. We use both a Gaussian prior (left) and horseshoe prior (right). Each point is a dataset: its x-axis coordinate denotes the median time for NUTS, and its y-axis coordinate denotes the median time for CGGibbs. The region beyond the dashed lines indicates that the corresponding sampler did not reach a minimum of 100 ESS within three days.

Finally, combining Eq. (7), Eq. (11) and Theorems 4 and 6 in Roberts and Sahu (1997) gives us

$$\rho(B(\Sigma)) = \rho_{L^2}(DUGS, \Sigma) \le \rho_{L^2}(RSGS, \Sigma) \le \exp\left(-\frac{1}{\kappa(\Sigma)}\right).$$

□

## B.2 Proof of Proposition 4.2

Let $\Sigma' := D\Sigma D$ be the covariance matrix after diagonal preconditioning using the diagonal matrix $D$ and set $Q' = (\Sigma')^{-1} = D^{-1}\Sigma^{-1}D^{-1}$.

$$\begin{aligned}
A(\Sigma') &= I - \mathrm{diag}((Q')_{11}^{-1}, (Q')_{22}^{-1}, \dots, (Q')_{dd}^{-1})Q' \\
&= DD^{-1} - D\,\mathrm{diag}(Q_{11}^{-1}, Q_{22}^{-1}, \dots, Q_{dd}^{-1})D^{-1}DQD^{-1} \\
&= D(I - \mathrm{diag}(Q_{11}^{-1}, Q_{22}^{-1}, \dots, Q_{dd}^{-1})Q)D^{-1} \\
&= DA(\Sigma)D^{-1} \\
&= D(U(\Sigma) + L(\Sigma))D^{-1} \\
&= DU(\Sigma)D^{-1} + DL(\Sigma)D^{-1}.
\end{aligned}$$

Since $DU(\Sigma)D^{-1}$ and $DL(\Sigma)D^{-1}$ are upper and lower triangular matrices, respectively, we get

$$U(\Sigma') = DU(\Sigma)D^{-1}, \quad L(\Sigma') = DL(\Sigma)D^{-1}.$$

Next, we have

$$\begin{aligned}
B(\Sigma') &= (I - L(\Sigma'))^{-1}U(\Sigma') \\
&= (DD^{-1} - DL(\Sigma)D^{-1})^{-1}DU(\Sigma)D^{-1} \\
&= D(I - L(\Sigma))^{-1}U(\Sigma)D^{-1} \\
&= DB(\Sigma)D^{-1}.
\end{aligned}$$

Now, let $\lambda$ and $x$ be an eigenvalue and eigenvector of $B(\Sigma')$, then

$$\begin{aligned}
&B(\Sigma')x = \lambda x \\
\Leftrightarrow\, &DB(\Sigma)D^{-1}x = \lambda x \\
\Leftrightarrow\, &B(\Sigma)(D^{-1}x) = D^{-1}\lambda x.
\end{aligned}$$

Therefore, $D^{-1}x$ is an eigenvector of $B(\Sigma)$ and $\lambda$ is an eigenvalue of $B(\Sigma)$ so that any eigenvalue of $B(\Sigma')$ is an eigenvalue of $B(\Sigma)$. Similarly, we also have any eigenvalue of $B(\Sigma)$ is an eigenvalue of $B(\Sigma')$. Hence, $B(\Sigma)$ and $B(\Sigma')$ have the same set of eigenvalues which means they also have the same maximum eigenvalue modulus. That is, $\rho(B(\Sigma)) = \rho(B(\Sigma'))$.

## C From theory to practice

In this section, we provide additional details on Section 4 and make the connection from theoretical results to experimental results.

### C.1 Divergences between distributions

Let $\pi_1, \pi_2$ be given distributions and let $\mathcal{C}(\pi_1, \pi_2)$ be the set of all couplings of $\pi_1$ and $\pi_2$ (joint distributions admitting $\pi_1$ and $\pi_2$ as marginals). The divergences used to study the convergence of samplers to their target distributions in most works include the total variation (TV) distance

$$\mathrm{TV}(\pi_1, \pi_2) = \inf_{(X,Y)\in\mathcal{C}(\pi_1,\pi_2)} \mathbb{P}(X \neq Y),$$

the $p$-Wasserstein ($\mathrm{W}_p$) ($p \geq 1$) distance

$$\mathrm{W}_p(\pi_1, \pi_2) = \left( \inf_{(X,Y)\in\mathcal{C}(\pi_1,\pi_2)} \mathbb{E}(\|X - Y\|^p) \right)^{1/p},$$

and the Pearson-$\chi^2$ divergence

$$\chi^2(\pi_1, \pi_2) = \mathbb{E}\left[ \left( \frac{\pi_1(X) - \pi_2(X)}{\pi_2(X)} \right)^2 \right], \quad X \sim \pi_2.$$

More generally, we can define the Pearson-$\chi^2$ divergence for any $\pi_1 \ll \pi_2$ as

$$\chi^2(\pi_1, \pi_2) = \int \left| \frac{\mathrm{d}\pi_1}{\mathrm{d}\pi_2} - 1 \right|^2 \mathrm{d}\pi_2.$$

The divergences presented above have several properties. We state some here without proof. For instance,

$$\mathrm{W}_p(\pi_1, \pi_2) \geq \mathrm{W}_q(\pi_1, \pi_2), \qquad p \geq q \geq 1,$$

and

$$\mathrm{TV}(\pi_1, \pi_2) \leq \frac{1}{2}\sqrt{\chi^2(\pi_1, \pi_2)}.$$

### C.2 Convergence rates and mixing times

Another quantity of interest is the mixing time of a sampler, which is defined by the number of iterations required for a sampler to be within a certain divergence of its target distribution. Precisely, the *mixing time* of a Markov kernel $K(\cdot, \cdot)$ with initial distribution $\mu_0$ and target distribution $\pi$, with respect to a divergence D, is defined as:

$$t^K_{\mathrm{mix}}(\mathrm{D}, \epsilon, \mu_0, \pi) := \inf\{t : \mathrm{D}(K^t\mu_0, \pi) \leq \epsilon\}, \quad \epsilon > 0.$$

The *mixing rate* is often used to compare the scaling of a certain quantity. Depending on which quantity that is of interest (e.g., $d$ or $\kappa$), we can throw away low order terms in the mixing time with respect to that quantity. For instance, a mixing time of $O(d \log d)$ correspond to mixing rate $O(d)$ if we focus on $d$ scaling.

An important distinction between convergence rate and mixing time is that convergence rate is asymptotic with respect to the number of iterations $t$, which does not capture its precise dependence on $d$ and $\kappa$. The main reason for this is due to the possible dependency on $d$ and $\kappa$ of the constant in front of the convergence rate. Therefore, a tight mixing time bound can be a more comprehensive metric of performance for a sampler of interest.

## C.3 From convergence rate to ESS

Throughout this paper, we have relied on the effective sample size (ESS) to evaluate the performance of different MCMC samplers. On the other hand, the theory we have discussed has mainly focused on mixing rates or times of MCMC algorithms with respect to different divergences. In this section, we provide two results that connect the theoretical convergence rate in total variation (TV) and $\chi^2$-divergence to the ESS.

Let us begin by introducing a few concepts. We say that a $\pi$-invariant Markov kernel $K$ and test function $f \in L^2(\pi)$ satisfy a CLT if there exists $0 < \sigma_f^2 < \infty$ such that

$$\sqrt{T}\left(\frac{1}{T}\sum_{t=0}^{T-1} f(X_t) - \pi(f)\right) \xrightarrow{d} \mathcal{N}(0, \sigma_f^2), \quad \text{as } T \to \infty. \tag{12}$$

See Roberts and Rosenthal (2004) for a discussion of necessary and sufficient conditions. When Eq. (12) holds, the variance of the running average satisfies (see e.g. Song and Schmeiser, 1995, Prop. 1)

$$\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1} f(X_t)\right) = \frac{\sigma_f^2}{T} + O(T^{-2}). \tag{13}$$

To see the importance of this equation, let us first simplify the notation (without loosing generality) by concentrating on the subset of zero-mean test functions $f \in L_0^2(\pi) := \{f \in L^2(\pi) : \pi(f) = 0\}$. Then Eq. (13) can be rewritten as

$$\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1} f(X_t)\right) = \frac{\pi(f^2)}{T_{\text{eff}}(f)} + O(T^{-2}). \tag{14}$$

where

$$T_{\text{eff}}(f) := T\frac{\pi(f^2)}{\sigma_f^2},$$

is the ESS of the test function $f$. Eq. (14) shows that, asymptotically in $T$, the variance of the running average $T^{-1}\sum_{t=0}^{T-1} f(X_t)$ is approximately equal to the variance achieved by a simple Monte Carlo estimator based on roughly $T_{\text{eff}}(f)$ i.i.d. samples from $\pi$. In particular, $T_{\text{eff}}(f) = T$ for every $f$ in the ideal case where $K$ produces exact i.i.d. samples from $\pi$—that is, when $K(x, \cdot) = \pi$ for $\pi$-almost every $x$.

In the following we rely on the fact that, when Eq. (12) holds, $\sigma_f^2$ can be written as (Roberts and Rosenthal, 2004)

$$\sigma_f^2 = \pi(f^2) + 2\sum_{t=1}^{\infty} \mathrm{Cov}(f(X_0), f(X_t)),$$

with $X_0 \sim \pi$ and $X_{t+1}|X_t \sim K(X_t, \cdot)$ for all $t \geq 0$.

**Connection between TV and ESS** The following result shows that for uniformly ergodic Markov chains with rate $\rho \in (0, 1)$, the ESS of a certain class of functions admits a uniform lower bound that is decreasing in $\rho$. In other words, a more efficient sampler—that is, one with $\rho$ closer to 0—should produce a uniformly higher ESS for a certain class of test functions.

**Proposition C.1.** *Suppose that there exists a constant $C \geq 0$ and $\rho \in (0, 1)$ such that for $\pi$-almost every $x \in \mathcal{X}$ we have*

$$\mathrm{TV}(K^t(x, \cdot), \pi) \leq C\rho^t.$$

*Then, for any test function $f \in L_0^2(\pi)$ with $|f| \leq 1$, we have that*

$$T_{\text{eff}}(f) \geq \frac{T}{1 + \frac{4C}{\pi(f^2)}\frac{\rho}{(1-\rho)}}.$$

*Proof of Proposition C.1.* Recall that the total variation distance between two probability measures $\mu$ and $\nu$ can be characterized as (see e.g. Roberts and Rosenthal, 2004, Prop 3(b))

$$\mathrm{TV}(\mu, \nu) = \frac{1}{2} \sup_{|f| \leq 1} |\mu(f) - \nu(f)|,$$

where the supremum is taken over all bounded functions $f : \mathcal{X} \to [-1, 1]$. Hence, for any $f \in L_0^2(\pi)$ with $|f| \leq 1$ we have

$$|K^t f(x)| \leq 2C\rho^t, \quad \pi - a.e.\ x.$$

Then, using the Markov property, Jensen's inequality, and the fact that $|f| \leq 1$, we obtain

$$|\mathrm{Cov}(f(X_0), f(X_t))| = |\mathbb{E}[f(X_0)f(X_t)]| = |\pi(fK^t f)| \leq \pi(|f||K^t f|) \leq \pi(|K^t f|) \leq 2C\rho^t.$$

Consequently,

$$\sum_{t=1}^{\infty} \pi(fK^t f) \leq \frac{2C\rho}{1-\rho},$$

from which the result follows immediately. $\square$

Suppose now that we consider not one target distribution $\pi$ but a sequence $\pi_1, \pi_2, \dots$ of increasing complexity. For example, they might involve increasing dimensionality $d_i$ and/or condition number $\kappa_i$. In such case, if we have

$$\mathrm{TV}(K^t(x, \cdot), \pi_i) \leq C\rho_i^t,$$

then we can make predictions on the decay of the relative effective sample size as $i \to \infty$. To do so in an interpretable way, we first rewrite $\rho_i$ as

$$\rho_i = 1 - \frac{1}{z_i},$$

where $z_i \to \infty$ is a measure of complexity. For instance, suppose $z_i = i^{1/4}$ for optimally tuned and Metropolized HMC when targeting an $i$-dimensional isotropic normal distribution $\pi_i$.

**Corollary C.2.** *Suppose that there exists a constant $C \geq 0$ and $\rho_i \in (0, 1)$ such that for $\pi_i$-almost every $x \in \mathcal{X}$ we have*

$$\mathrm{TV}(K^t(x, \cdot), \pi_i) \leq C\rho_i^t,$$

*where $\rho_i = 1 - \frac{1}{z_i}$ and $z_i \to \infty$. Then, for any sequence of test functions $\{f_i\}_{i \geq 1}$ with each $f_i \in L_0^2(\pi_i)$, $|f_i| \leq 1$, and $\limsup_{i \to \infty} \pi_i(f_i^2) < \infty$, we have*

$$\textit{relative ESS} = \frac{T_{\text{eff}}(f_i)}{T} \geq b_i \sim \frac{\tilde{C}_i}{z_i}, \quad i \to \infty,$$

*where $\tilde{C}_i = \pi_i(f_i^2)/(4C)$.*

*Proof of Corollary C.2.* From Proposition C.1, we can pick

$$b_i = \left(1 + \frac{4C}{\pi_i(f_i^2)} \frac{\rho_i}{(1-\rho_i)}\right)^{-1}.$$

From our assumptions on $z_i, \rho_i$, we have $\rho_i/(1-\rho_i) = (1 - 1/z_i)/(1/z_i) = z_i - 1$. Let $\tilde{C}_i = \pi_i(f_i^2)/(4C)$. We can rewrite the above expression as

$$b_i = \left(1 + \frac{1}{\tilde{C}_i}(z_i - 1)\right)^{-1}.$$

It remains to show that $b_i \sim \tilde{C}_i/z_i$ as $i \to \infty$:

$$\lim_{i\to\infty} \frac{\tilde{C}_i/z_i}{b_i} = \lim_{i\to\infty} \frac{\tilde{C}_i}{z_i}\left(1 + \frac{1}{\tilde{C}_i}(z_i - 1)\right) = \lim_{i\to\infty}\left(\frac{\tilde{C}_i}{z_i} + 1 - \frac{1}{z_i}\right) = 1.$$

□

For example, in the case of optimally tuned Metropolized HMC on isotropic normal targets, if $z_i = i^{1/4}$, we obtain from the above corollary that the relative ESS decays at the relatively slow rate of at most $1/z_i = i^{-1/4}$.

**Connection between $\chi^2$ and ESS** For a reversible Markov kernel $K$, Proposition C.4 below allows us to translate the convergence rate in $\chi$-squared divergence into a uniform lower bound on the ESS of $L_2(\pi)$ functions.

We start with a key technical lemma that draws the connection between the convergence rate in $\chi$-squared divergence and the decay of variance for $f \in L_2^0(\pi)$.

**Lemma C.3.** *Suppose the Markov kernel $K$ is reversible with respect to $\pi$. Further suppose that there exists $\rho \in [0, 1)$ such that for any initial distribution $\pi_0 \ll \pi$ satisfying $\chi^2(\pi_0, \pi) < \infty$ and for all $t \geq 0$, we have*

$$\chi^2(\pi_0 K^t, \pi) \leq \rho^t \chi^2(\pi_0, \pi).$$

*Then, for all $f \in L_2^0(\pi)$ and for all $t > 0$, we have*

$$\|K^t f\|^2_{L_2(\pi)} \leq \rho^t \|f\|^2_{L_2(\pi)}.$$

*Proof of Lemma C.3.* First, by the reversibility of $K$ (and $K^t$ for all $t > 0$) with respect to $\pi$, we have that for all distributions $\pi_0$ such that $\chi^2(\pi_0, \pi) < \infty$,

$$K^t \frac{\mathrm{d}\pi_0}{\mathrm{d}\pi} = \frac{\mathrm{d}\pi_0 K^t}{\mathrm{d}\pi} \qquad (\pi\text{-a.e.}). \tag{15}$$

To see this, we observe that for all $g \in L_2(\pi)$,

$$\begin{aligned}\int K^t\left(\frac{\mathrm{d}\pi_0}{\mathrm{d}\pi}\right) g \mathrm{d}\pi &= \int \frac{\mathrm{d}\pi_0}{\mathrm{d}\pi} K^t(g)\mathrm{d}\pi \quad \text{(by the reversibility of } K^t)\\ &= \int K^t g \mathrm{d}\pi_0\\ &= \pi_0 K^t(g)\\ &= \int \frac{\mathrm{d}\pi_0 K^t}{\mathrm{d}\pi} g \mathrm{d}\pi.\end{aligned}$$

Eq. (15) combined with the definition of the $\chi$-squared divergence then allows us to interpret $\chi^2(\pi_0 K^t, \pi) \leq \rho^t \chi^2(\pi_0, \pi)$ as

$$\left\| K^t\left(\frac{\mathrm{d}\pi_0}{\mathrm{d}\pi} - 1\right)\right\|^2_{L_2(\pi)} \leq \rho^t \left\|\frac{\mathrm{d}\pi_0}{\mathrm{d}\pi} - 1\right\|^2_{L_2(\pi)}. \tag{16}$$

Under the assumption of Lemma C.3, we have that Eq. (16) holds for all $t$ and for all $\pi_0 \ll \pi$ such that $\chi^2(\pi_0, \pi) < \infty$.

Consequently, we can conclude that for all bounded $f \in L_2^0(\pi)$ (i.e., $\sup_x |f(x)| < \infty$) and for all $t > 0$, we have

$$\|K^t f\|^2_{L_2(\pi)} \leq \rho^t \|f\|^2_{L_2(\pi)}. \tag{17}$$

To see this, define the measure

$$p_f(A) = \int_A \frac{f}{\sup_x |f(x)|} + 1 \, \mathrm{d}\pi,$$

which satisfies $p_f \ll \pi$ and $\chi^2(p_f, \pi) < \infty$. Then,

$$\frac{\mathrm{d}p_f}{\mathrm{d}\pi} = \frac{f}{\sup_x |f(x)|} + 1,$$

and applying Eq. (16) with $p_f$ in place of $\pi_0$, we get Eq. (17).

Finally, we focus on removing the boundedness assumption on $f$. For any $f \in L_2^0(\pi)$, define

$$f_{L,M}(x) := \begin{cases} \min\{f(x), M\} & \text{as } x \geq 0 \\ \max\{f(x), -L\} & \text{as } x < 0 \end{cases}$$

Notice that for any $M > 0$, there exists $L > 0$ such that $\pi(f_{L,M}) = 0$.

Consider an increasing sequence $M_n$ such that $\lim_{n\to\infty} M_n = \infty$. For any $f \in L_2^0(\pi)$, we can construct an approximating sequence $f_n := f_{L_n,M_n}$, where $L_n$ is chosen so that for all $n \in \mathbb{N}$, $f_{L_n,M_n} \in L_2^0(\pi)$. Notably, $L_n$ has to be a non-decreasing sequence, $\lim_{n\to\infty} \|f_n - f\|_{L_2(\pi)} = 0$, and $\sup_x f_{L_n,M_n} = \max\{L_n, M_n\} < \infty$. Therefore, for all $f \in L_2^0(\pi)$ and for all $n \in \mathbb{N}$, we have that

$$\|K^t f_n\|^2_{L_2(\pi)} \leq \rho^t \|f_n\|^2_{L_2(\pi)}, \quad \text{for all } t > 0. \tag{18}$$

Because $\|f_n\|_{L_2(\pi)}$ converges increasingly to $\|f\|_{L_2(\pi)}$ as $n \to \infty$, we invoke the monotone convergence theorem to take the limit $n \to \infty$ on both sides of Eq. (18), which completes the proof. □

**Proposition C.4.** *Under the same conditions of Lemma C.3, we have that*

$$\inf_{f \in L_0^2(\pi)} T_{\text{eff}}(f) \geq \frac{T}{1 + 2\frac{\sqrt{\rho}}{1-\sqrt{\rho}}}.$$

*Proof of Proposition C.4.* Lemma C.3 yields that for all $f \in L_2^0(\pi)$ and all $t \geq 0$,

$$\pi((K^t f)^2) \leq \rho^t \pi(f^2).$$

Therefore, by the Cauchy–Schwarz inequality,

$$|\pi(f K^t f)| \leq \sqrt{\pi(f^2)}\sqrt{\pi((K^t f)^2)} \leq \pi(f^2)(\sqrt{\rho})^t.$$

Hence,

$$\sum_{t=1}^{\infty} \pi(f K^t f) \leq \pi(f^2)\frac{\sqrt{\rho}}{1-\sqrt{\rho}},$$

and the claim then follows by replacing the above in the definition of $T_{\text{eff}}(f)$ and then taking the infimum. □

We provide an analogous result to Corollary C.2 for a reversible kernel $K$ in terms of $\chi$-squared divergence convergence.

**Corollary C.5.** *Consider a sequence of target distributions $\pi_i$. Suppose there exists $\rho_i \in (0,1)$ such that for all $i$, for all $t > 0$, and for all initial distributions $\mu_{0,i}$ satisfying $\chi^2(\mu_{0,i}, \pi_i) < \infty$, we have*

$$\chi^2(\mu_{0,i}K^t, \pi_i) \leq \rho_i^t \chi^2(\mu_{0,i}, \pi_i),$$

*where $\rho_i = 1 - \frac{1}{z_i}$ and $z_i \to \infty$. Then, for any sequence of test functions $\{f_i\}_{i\geq 1}$ with each $f_i \in L_0^2(\pi_i)$, we have*

$$\textit{relative ESS} = \frac{T_{\text{eff}}(f_i)}{T} \geq \frac{1}{1 + 2\frac{\sqrt{\rho_i}}{1-\sqrt{\rho_i}}} \sim \frac{1}{4z_i}, \; i \to \infty.$$

*Proof of Corollary C.5.* We only need to show that $\frac{1}{1+2\frac{\sqrt{\rho_i}}{1-\sqrt{\rho_i}}} \sim \frac{1}{4z_i}$. We do this by showing that

$$\frac{\sqrt{\rho_i}}{1-\sqrt{\rho_i}} \sim \frac{1}{1-\sqrt{\rho_i}} \sim 2z_i.$$

Because $\rho_i = 1 - 1/z_i$ and $z_i \to \infty$, we have $\rho_i \to 1$ and so the first asymptotic equivalence follows. Next, by a Taylor expansion of $h(x) = \sqrt{1-1/x}$ about $x = \infty$, we have $h(x) = 1 - 1/(2x) + o(1/x)$. Therefore, $1 - \sqrt{\rho_i} \to 1/(2z_i)$. □

**Remark C.6.** *Note that the bounds in Propositions C.1 and C.4 should be reasonably tight in the limit $\rho \to 0$, since for $\rho = 0$ they both recover the i.i.d. case $T_{\text{eff}}(f) = T$.*

### C.4 From Gibbs to Metropolis-within-Gibbs

While the theoretical results discussed in this section concern Gibbs samplers (i.e., using full conditional updates), the methodology described in Section 3 applies to any "Metropolis-within-Gibbs" algorithm. The reason for this gap is that the theoretical analysis of the Gibbs sampler is more developed than that of Metropolis-within-Gibbs samplers. Fortunately, recent work has shown how to relate the convergence properties of Metropolis-within-Gibbs samplers with their idealized, full conditional counterparts (Ascolani et al., 2024b). Invariance under axis-aligned stretching does not hold for Metropolis-within-Gibbs samplers. However, algorithms such as the slice sampler with doubling are expected to be relatively insensitive to axis-aligned transformations (Neal, 2003).

## D Connection between graphical model and compute graph

In this section, we sketch a simple proof that all savings made from graphical models derived Markov blankets are recovered as special cases of the cached compute graph method.

Consider a target distribution $\pi$ in $\mathbb{R}^d$ and, for any node $\theta_i$ in the graphical model of $\pi$, denote the parent nodes of $\theta_i$ by $\mathcal{P}_i(\theta)$. Then, for all $\theta = (\theta_1, \ldots, \theta_d) \in \mathbb{R}^d$,

$$\pi(\theta) = \prod_{i=1}^{d} p_i(\theta_i | \mathcal{P}_i(\theta)).$$

Now rewriting $p_i(\theta_i|\mathcal{P}_i(\theta)) = \exp f_i(S_i)$ where $S_i$ is the concatenation of $\theta_i$ with $\mathcal{P}_i(\theta)$, we have

$$\log \pi(\theta) = \sum_{i=1}^{d} f_i(S_i).$$

This last expression encodes the factor graph associated with the directed graphical model. To construct the compute graph of $\log \pi$ from its factor graph, we create a new node corresponding to the output $\log \pi(\theta)$ and connect all $f_i(S_i)$ nodes to this new node via a summation node. Crucially, that summation node is cached in the same way as described in Section 3, i.e., by subtracting and adding updated values. Finally, by expanding the nodes $f_i(S_i)$ nodes into their respective compute graphs, we obtain the compute graph for $\log \pi$.

Notice that when a node $\theta_i$ is updated, only the $f_j$'s connected to $\theta_i$ need to be updated. This set of $f_j$'s coincide with the Markov blanket. Moreover, the number of updates to the summation node is equal to the number of $f_j$'s updated.

## E Effects of Gibbs sweep

In this section, we investigate how different types of Gibbs samplers affect the time per ESS of the chain. Here, we consider 3 types of Gibbs samplers: DUGS, RSGS and random permutation Gibbs sampler (RPGS), where a random but fixed update order is generated each sweep. Next, we repeat the experiment in Fig. 4 with both normal and horseshoe priors for the 3 Gibbs samplers. The results are shown in Fig. 18 and Fig. 19. We can see from these figures that the scalings in time per ESS for all 3 Gibbs samplers are relatively similar. This justifies our decision to only use DUGS as the representative for Gibbs samplers. Moreover, DUGS tends to have the best

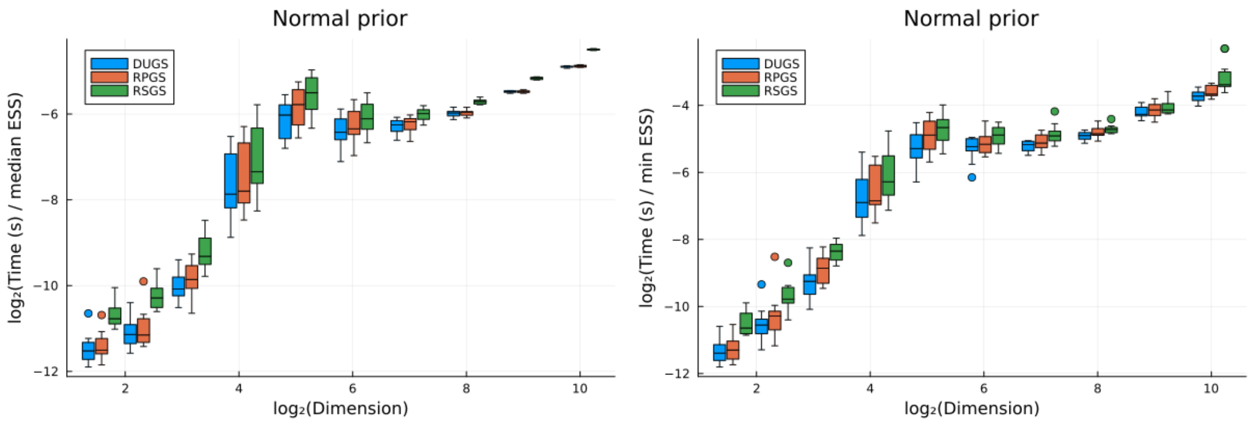


Figure 18: Time per ESS for CGGibbs a function of dimension for colon dataset with normal priors. The box plots summarize the results over 10 replicates.

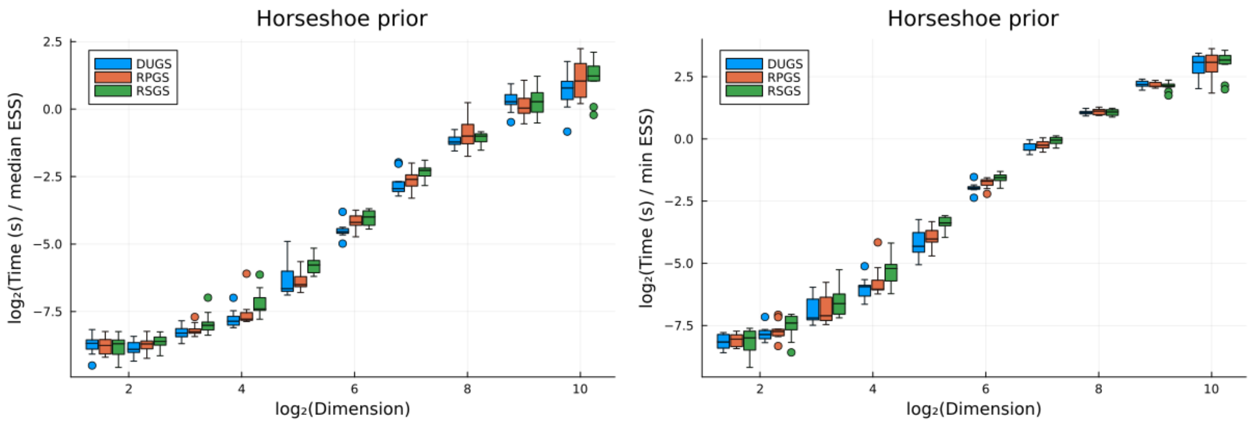


Figure 19: Time per ESS for CGGibbs a function of dimension for colon dataset with horseshoe priors. The box plots summarize the results over 10 replicates.

time per ESS, RSGS the worst and RPGS somewhere in between, further justifying our choice to use DUGS. This is somewhat expected since, for normal targets, we often have $\rho_{L^2}(DUGS) \leq \rho_{L^2}(RSGS)$ (Roberts and Sahu, 1997) and RPGS is intuitively a mix of DUGS and RSGS.