Automatic Reparameterisation of Probabilistic Programs

Maria I. Gorinova [*1] **Dave Moore** [2] **Matthew D. Hoffman** [2]

Abstract

Probabilistic programming has emerged as a powerful paradigm in statistics, applied science, and machine learning: by decoupling modelling from inference, it promises to allow modellers to directly reason about the processes generating data. However, the performance of inference algorithms can be dramatically affected by the parameterisation used to express a model, requiring users to transform their programs in non-intuitive ways. We argue for automating these transformations, and demonstrate that mechanisms available in recent modelling frameworks can implement non-centring and related reparameterisations. This enables new inference algorithms, and we propose two: a simple approach using interleaved sampling and a novel variational formulation that searches over a continuous space of parameterisations. We show that these approaches enable robust inference across a range of models, and can yield more efficient samplers than the best fixed parameterisation.

1. Introduction

Reparameterising a probabilistic model means expressing it in terms of new variables defined by a bijective transformation of the original variables of interest. The reparameterised model expresses the same statistical assumptions as the original, but can have drastically different posterior geometry, with significant implications for both variational and sampling-based inference algorithms.

Non-centring is a particularly common form of reparameterisation in Bayesian hierarchical models. Consider a random variable $z \sim \mathcal{N}(\mu, \sigma)$; we say this is in *centred* parameterisation (CP). If we instead work with an auxiliary, standard normal variable $\tilde{z} \sim \mathcal{N}(0, 1)$, and obtain z by applying the transformation $z = \mu + \sigma\tilde{z}$, we say the variable $\tilde{z}$ is in its *non-centred* parameterisation (NCP). Although the centred parameterisation is often more intuitive, non-centring can dramatically improve the performance of inference (Betancourt & Girolami, 2015). *Neal's funnel* (Figure 1a) provides a simple example: most Markov chain Monte Carlo (MCMC) algorithms have trouble sampling from the funnel due to the strong non-linear dependence between latent variables. Non-centring the model removes this dependence, converting the funnel into a spherical Gaussian distribution.

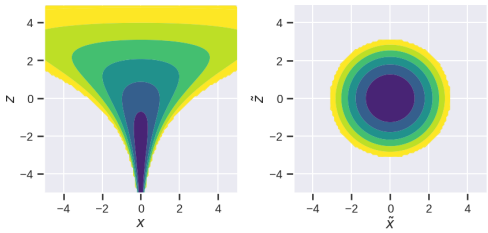
Bayesian practitioners are often advised to manually non-centre their models (Stan Development Team et al., 2016); however, this breaks the separation between modelling and inference and requires expressing the model in a potentially less intuitive form. Moreover, it requires the user to understand the concept of non-centring and to know *a priori* where in the model it might be appropriate. Because the best parameterisation for a given model may vary across datasets, even experts may need to find the optimal parameterisation by trial and error, burdening modellers and slowing down the model development loop (Blei, 2014).

We propose that non-centring and similar reparameterisations be handled automatically by probabilistic programming systems. We demonstrate how such program transformations may be implemented using the effect handling mechanisms present in several modern deep probabilistic programming frameworks, and consider two inference algorithms enabled by automatic reparameterisation: interleaved Hamiltonian Monte Carlo (iHMC), which alternates HMC steps between centred and non-centred parameterisations, and a novel algorithm we call Variationally Inferred Parameterisation (VIP), which searches over a continuous space of reparameterisations that includes non-centring as a special case.[1] We compare these strategies to a fixed centred and non-centred parameterisation across a range of well-known hierarchical models. Our results suggest that both VIP and iHMC can enable for more automated robust inference, often performing at least as well as the best fixed parame-

[*]Work done while interning at Google. [1]University of Edinburgh, Edinburgh, UK [2]Google, San Francisco, CA, USA. Correspondence to: Maria I. Gorinova <m.gorinova@ed.ac.uk>, Dave Moore <davmre@google.com>, Matthew D. Hoffman <mhoffman@google.com>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

[1]Code for these algorithms and experiments is available at https://github.com/mgorinova/autoreparam.



(a) Centred (left) and non-centred (right) parameterisation.

$$\begin{aligned}&\text{NealsFunnel}(z,\ x):\\ &\quad z \sim \mathcal{N}(0,3)\\ &\quad x \sim \mathcal{N}(0, \exp(z/2))\end{aligned}$$

(b) Model that generates variables z and x.

$$\begin{aligned}&z = 0\\ &\text{lp}_z = \log p_{\mathcal{N}}(z \mid 0, 3)\\ &x = 0\\ &\text{lp}_x = \log p_{\mathcal{N}}(x \mid 0, \exp(z/2))\end{aligned}$$

(c) The model in the context of `log_prob_at_0`.

Figure 1. Neal's funnel (Neal, 2003): $z \sim N(0, 3)$; $x \sim N(0, e^{z/2})$.

terisation and sometimes better, without requiring *a priori* knowledge of the optimal parameterisation. Both strategies have the potential to free modellers from thinking about manual reparameterisation, accelerate the modelling cycle, and improve the robustness of inference in next-generation modelling frameworks.

2. Related Work

The value of non-centring is well-known to MCMC practitioners and researchers (Stan Development Team et al., 2016; Betancourt & Girolami, 2015), and can also lead to better variational fits in hierarchical models (Yao et al., 2018). However, the literature largely treats this as a modelling choice; Yao et al. (2018) propose that "there is no general rule to determine whether non-centred parameterisation is better than the centred one." We are not aware of prior work that treats non-centring directly as a computational phenomenon to be exploited by inference systems.

Non-centred parameterisation of probabilistic models can be seen as analogous to the reparameterisation trick in stochastic optimisation (Kingma & Welling, 2013); both involve expressing a variable in terms of a diffeomorphic transformation from a "standardised" variable. In the context of probabilistic inference, these are complementary tools: the reparameterisation trick yields low-variance stochastic gradients of variational objectives, whereas non-centring changes the geometry of the posterior itself, leading to qualitatively different variational fits and MCMC trajectories.

In the context of Gibbs sampling, Papaspiliopoulos et al. (2007) introduce a family of *partially non-centred* parameterisations similar to those we use in VIP (described below) and show that it improves mixing in a spatial GLMM. Our current work can be viewed as an general-purpose extension of this work that mechanically reparameterises user-provided models and automates the choice of parameterisation. Similarly, Yu & Meng (2011) proposed a Gibbs sampling scheme that interleaves steps in centred and non-centred parameterisations; our interleaved HMC algorithm can be viewed as an automated, gradient-based descendent of their scheme.

Recently, there has been work on accelerating MCMC inference through learned reparameterisation: Parno & Marzouk (2018) and Hoffman et al. (2019) run samplers in the image of a bijective map fitted to transform the target distribution approximately to an isotropic Gaussian. These may be viewed as 'black-box' methods that rely on learning the target geometry, potentially using highly expressive neural variational models, while we use probabilistic-program transformations to apply 'white-box' reparameterisations similar to those a modeller could in principle implement themselves. Because they exploit model structure, white-box approaches can correct pathologies such as those of Neal's funnel (Figure 1a) directly, reliably, and at much lower cost (in parameters and inference overhead) than black-box models. White- and black-box reparameterisations are not mutually exclusive, and may have complementary advantages; combining them is a likely fruitful direction for improving inference in structured models.

Previous work in probabilistic programming has been exploring other 'white-box' approaches to perform or optimise inference. For example, Hakaru (Narayanan et al., 2016; Zinkov & Shan, 2017) and PSI (Gehr et al., 2016; 2020) use program transformations to perform symbolic inference, while Gen (Cusumano-Towner et al., 2019) and SlicStan (Gorinova et al., 2019) can statically analyse the model structure to compile to a more efficient inference strategy. To the best of our knowledge, the approach presented in this paper is the first to apply variational inference as a dynamic pre-processing step, which optimises the program based on both the program structure and observed data.

3. Understanding the Effect of Reparameterisation

Non-centring reparameterisation is not always optimal; its usefulness depends on properties of both the model and the observed data. In this section, we develop intuition by working with a simple hierarchical model for which we can derive the posterior analytically. Consider a simple

realisation of a model discussed by Betancourt & Girolami (2015, (2)), where for a vector of N datapoints $\mathbf{y}$, and some given constants σ and σ_μ, we have:

$$\theta \sim \mathcal{N}(0,1) \qquad \mu \sim \mathcal{N}(\theta, \sigma_\mu)$$
$$y_n \sim \mathcal{N}(\mu, \sigma) \;\text{ for all } n \in 1 \ldots N$$

In the non-centred model, $\mathbf{y}$ is defined in terms of $\tilde{\mu}$ and θ, where $\tilde{\mu}$ is a standard Gaussian variable:

$$\theta \sim \mathcal{N}(0,1) \qquad \tilde{\mu} \sim \mathcal{N}(0,1)$$
$$y_n \sim \mathcal{N}(\theta + \sigma_\mu \tilde{\mu}, \sigma) \;\text{ for all } n \in 1 \ldots N$$

Figure 2a and Figure 2b show the graphical models for the two parameterisations. In the non-centred case, the direct dependency between θ and μ is substituted by a conditional dependency given the data $\mathbf{y}$, which creates an "explaining away" effect. Intuitively, this means that the stronger the evidence $\mathbf{y}$ is (large N, and small variance), the stronger the dependency between θ and $\tilde{\mu}$ becomes, creating a poorly-conditioned posterior that may slow inference.

As the Gaussian distribution is self-conjugate, the posterior in each case (centred or non-centred) is also a Gaussian distribution, and we can analytically inspect its covariance matrix V. To quantify the quality of the parameterisation in each case, we investigate the condition number κ of the posterior covariance matrix under the optimal diagonal preconditioner. This models the common practice (implemented in tools such as PyMC3 and Stan and followed in our experiments) of sampling using a fitted diagonal preconditioner.

Figure 2c shows the condition numbers κ_{CP} and κ_{NCP} for each parameterisation as a function of $q = N/\sigma^2$; the full derivation is in Appendix A. This figure confirms the intuition that the non-centred parameterisation is better suited for situation when the evidence is weak, while strong evidence calls for centred parameterisation. In this example we can exactly determine the optimal parameterisation, since the model has only one variable that can be reparameterised and the posterior has a closed form. In more realistic settings, even experts cannot predict the optimal parameterisation for hierarchical models with many variables and groups of data, and the wrong choice can lead to poor conditioning, heavy tails or other pathological geometry.

4. Reparameterising Probabilistic Programs

An advantage of probabilistic programming is that the program itself provides a structured model representation, and we can explore model reparameterisation through the lens of program transformations. In this paper, we focus on transforming *generative probabilistic programs* where the program represents a sampling process describing how the data was generated from some unknown latent variables. Most probabilistic programming languages (PPLs) provide some mechanism for transforming a generative process into an inference program; our automatic reparameterisation approach is applicable to PPLs that transform generative programs using *effect handling*. This includes modern deep PPLs such as Pyro (Uber AI Labs, 2017) and Edward2 (Tran et al., 2018).

4.1. Effect Handling-based Probabilistic Programming

Consider a generative program, where running the program forward generates samples from the prior over latent variables and data. Effect handling-based PPLs treat generating a random variable within such a model as an *effectful operation* (an operation that is understood as having side effects) and provide ways for resolving this operation in the form of *effect handlers*, to allow for inference. For example, we often need to transform a statement that generates a random variable to a statement that evaluates some (log) density or mass function. We can implement this using an effect handler:

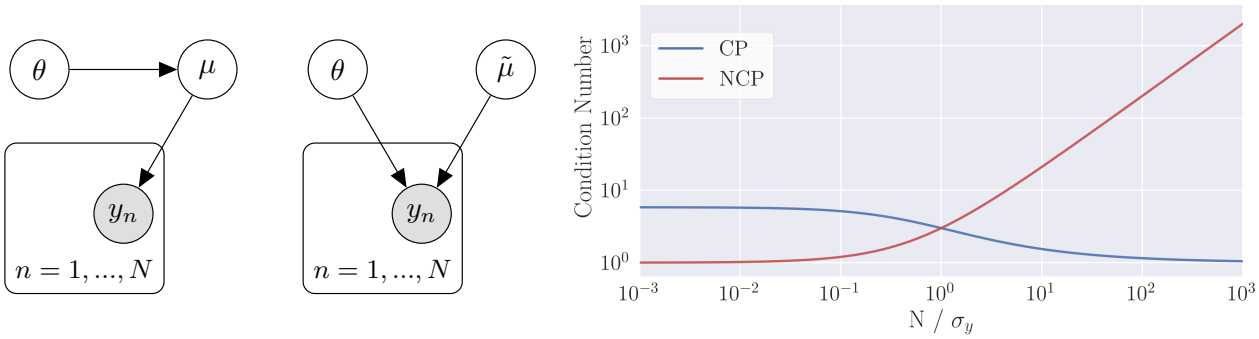
$$\texttt{log_prob_at_0} =$$
$$\quad \textbf{handler}\ \{v \sim \mathcal{D}(a_1, \ldots, a_N) \mapsto$$
$$\qquad\qquad v = 0; \mathrm{lp}_v = \log p_{\mathcal{D}}(v \mid a_1, \ldots, a_N)\}^2$$

The handler `log_prob_at_0` handles statements of the form $v \sim \mathcal{D}(a_1, \ldots, a_N)$. Such statements normally mean "sample a random variable from the distribution $\mathcal{D}(a_1, \ldots, a_N)$ and record its value in v". However, when executed in the context of `log_prob_at_0` (we write **`with`** `log_prob_at_0` **`handle`** `model`), statements that contain random-variable constructions are *handled* by setting the value of the variable v to 0, then evaluating the log density (or mass) function of $\mathcal{D}(a_1, \ldots, a_N)$ at $v = 0$ and recording its value in a new (program) variable lp_v.

For example, consider the function implementing Neal's funnel in Figure 1b. When executed without any context, this function generates two random variables, z and x. When executed in the context of the `log_prob_at_0` handler, it does not generate random variables, but it instead evaluates $\log p_{\mathcal{N}}(z \mid 0, 3)$ and $\log p_{\mathcal{N}}(x \mid 0, \exp(z/2))$ (Figure 1c).

This approach can be extended to produce a function that corresponds to the log joint density (or mass) function of the latent variables of the model. In §§ B.1, we give the pseudo-code implementation of a function `make_log_joint`, which takes a model $M(\mathbf{z} \mid \mathbf{x})$ — that generates latent variables $\mathbf{z}$ and generates and observes data $\mathbf{x}$ — and returns the function $f(\mathbf{z}) = \log p(\mathbf{z}, \mathbf{x})$. This is

[2] Algebraic effects and handlers typically involve passing a continuation within the handler. We make the continuation implicit to stay close to Edward2's implementation.



(a) Centred. (b) Non-centred. (c) The condition number as a function of the data's strength.

Figure 2. Effects of reparameterising a simple model with known posterior.

a core operation, as it transforms a generative model into a function proportional to the posterior distribution, which can be repeatedly evaluated and automatically differentiated to perform inference.

More generally, effectful operations are operations that can have side effects, e.g. writing to a file. The programming languages literature formalises cases where impure behaviour arises from a set of effectful operations in terms of algebraic effects and their handlers (Plotkin & Power, 2001; Plotkin & Pretnar, 2009; Pretnar, 2015). A concrete implementation for an effectful operation is given in the form of effect handlers, which (similarly to exception handlers) are responsible for resolving the operation. Effect handlers can be used as a powerful abstraction in probabilistic programming, and have been incorporated into recent frameworks such as Pyro and Edward2 (Moore & Gorinova, 2018).

4.2. Model Reparameterisation Using Effect Handlers

Once equipped with an effect handling-based PPL, we can easily construct handlers to perform many model transformations, including model reparameterisation.

Non-centring Handler. `ncp` = **`handler`** $\{$
$v \sim \mathcal{N}(\mu, \sigma), v \notin \text{data} \mapsto \tilde{v} \sim \mathcal{N}(0,1); v = \mu + \sigma\tilde{v}\}$

A non-centring handler can be used to non-centre all standardisable [3] latent variables in a model. The handler simply applies to statements of the form $v \sim \mathcal{N}(\mu, \sigma)$, where v is not a data variable, and transforms them to $\tilde{v} \sim \mathcal{N}(0,1)$, $v = \mu + \sigma\tilde{v}$. When nested within a `log_prob` handler (like the one from §§ 4.1), `log_prob` handles the transformed standard normal statement $\tilde{v} \sim \mathcal{N}(0,1)$. Thus, `make_log_joint` applied to a model in the `ncp` context returns the log joint function of the transformed variables $\tilde{\mathbf{z}}$ rather than the original variables $\mathbf{z}$.

For example, `make_log_joint`(NealsFunnel(z, x)) gives:

$$\log p(z, x) = \log \mathcal{N}(z \mid 0, 3) + \log \mathcal{N}(x \mid 0, \exp(z/2))$$

`make_log_joint(`**`with`** `ncp` **`handle`** NealsFunnel(z, x)`)` corresponds to the function:

$$\log p(\tilde{z}, \tilde{x}) = \log \mathcal{N}(\tilde{z} \mid 0, 1) + \log \mathcal{N}(\tilde{x} \mid 0, 1)$$

where $z = 3\tilde{z}$ and $x = \exp(z/2)\tilde{x}$.

This approach can easily be extended to other parameterisations, including partially centred parameterisations (as shown later in §§ 5.2), non-centring and whitening multivariate Gaussians, and transforming constrained variables to have unbounded support.

Edward2 Implementation. We implement reparameterisation handlers in Edward2, a deep PPL embedded in Python and TensorFlow (Tran et al., 2018). A model in Edward2 is a Python function that generates random variables. In the core of Edward2 is a special case of effect handling called *interception*. To obtain the joint density of a model, the language provides the function `make_log_joint_fn(model)`[4], which uses a `log_prob` interceptor (handler) as previously described.

We extend the usage of interception to treat sample statements in one parameterisation as sample statements in another parameterisation (similarly to the `ncp` handler above):

```
def noncentre(rv_fn, **d):
    # Assumes a location-scale family.
    rv_fn = ed.interceptable[5](rv_fn)
    rv_std = rv_fn(loc=0, scale=1)
    return d["loc"] + d["scale"] * rv_std
```

[3] We focus on Gaussian variables, but non-centring is broadly applicable, e.g. to the location-scale family and random variables that can be expressed as a bijective transformation $z = f_\theta(\tilde{z})$ of a "standardised" variable $\tilde{z}$.

[4] Corresponds to `make_log_joint(model)` in our example.

[5] Wrapping the constructor with `ed.interceptable` ensures that we can nest this interceptor in the context of other interceptors.

We use the interceptor by executing a model of interest within the interceptor's context (using Python's context managers). This overrides each random variable's constructor to construct a variable with location 0 and scale 1, and scale and shift that variable appropriately:

```
with ed.interception(noncentre):
   neals_funnel()
```

We present and explain in more detail all interceptors used for this work in Appendix B.

5. Automatic Model Reparameterisation

We introduce two inference strategies that exploit automatic reparameterisation: interleaved Hamiltonian Monte Carlo (iHMC), and the Variationally Inferred Parameterisation (VIP).

5.1. Interleaved Hamiltonian Monte Carlo

Automatic reparameterisation opens up the possibility of algorithms that exploit multiple parameterisations of a single model. We consider *interleaved Hamiltonian Monte Carlo (iHMC)*, which uses two HMC steps to produce each sample from the target distribution: the first step is made in CP, using the original model latent variables, while the second step is made in NCP, using the auxiliary standardised variables. Interleaving MCMC kernels across parameterisations has been explored in previous work on Gibbs sampling (Yu & Meng, 2011; Kastner & Frühwirth-Schnatter, 2014), which demonstrated that CP and NCP steps can be combined to achieve more robust and performant samplers. Our contribution is to make the interleaving automatic and model-agnostic: instead of requiring the user to write multiple versions of their model and a custom inference algorithm, we implement iHMC as a black-box inference algorithm for centred Edward2 models.

Algorithm 1 outlines iHMC. It takes a single centred model $M_{cp}(\mathbf{z} \mid \mathbf{x})$ that defines latent variables $\mathbf{z}$ and generates data $\mathbf{x}$. It uses the function `make_ncp` to automatically obtain a non-centred version of the model, $M_{ncp}(\tilde{\mathbf{z}} \mid \mathbf{x})$, which defines auxiliary variables $\tilde{\mathbf{z}}$ and function f, such that $\mathbf{z} = f(\tilde{\mathbf{z}})$.

5.2. Variationally Inferred Parameterisation

The best parameterisation for a given model may mix centred and non-centred representations for different variables. To efficiently search the space of reparameterisations, we propose the *variationally inferred parameterisation (VIP)* algorithm, which selects a parameterisation by gradient-based optimisation of a differentiable variational objective. VIP can be used as a pre-processing step to another inference algorithm; as it only changes the parameterisation of the model, MCMC methods applied to the learned parameterisation maintain their asymptotic guarantees.

Consider a model with latent variables $\mathbf{z}$. We introduce *parameterisation parameters* $\boldsymbol{\lambda} = (\lambda_i) \in [0, 1]$ for each variable z_i, and transform $z_i \sim \mathcal{N}(z_i \mid \mu_i, \sigma_i)$ by defining $\tilde{z}_i \sim \mathcal{N}(\lambda_i \mu_i, \sigma_i^{\lambda_i})$ and $z_i = \mu_i + \sigma_i^{1-\lambda_i}(\tilde{z}_i - \lambda_i \mu_i)$. This defines a continuous relaxation that includes NCP as the special case $\boldsymbol{\lambda} = 0$ and CP as $\boldsymbol{\lambda} = 1$. More generally, it supports a combinatorially large class of per-variable and partial centrings.

Example. Recall the example model from Section 3, which defines the joint density $p(\theta, \mu, \mathbf{y}) = \mathcal{N}(\theta \mid 0, 1) \times \mathcal{N}(\mu \mid \theta, \sigma_\mu) \times \mathcal{N}(\mathbf{y} \mid \mu, \sigma)$. Using the parameterisation above to reparameterise μ, we get:

$$p(\theta, \hat{\mu}, \mathbf{y}) = \mathcal{N}(\theta \mid 0, 1) \times \mathcal{N}(\hat{\mu} \mid \lambda\theta, \sigma_\mu^{\lambda}) \times \mathcal{N}(\mathbf{y} \mid \theta + \sigma_\mu^{1-\lambda}(\hat{\mu} - \lambda\theta), \sigma)$$

Similarly to before, we analytically derive an expression for the posterior under different values of λ. Figure 4 shows the condition number $\kappa(\lambda)$ of the diagonally preconditioned posterior, for different values of $q = N/\sigma^2$ with fixed prior scale $\sigma_\mu = 1$. As expected, when the data is weak ($q = 0.01$), setting the parameterisation parameter λ to be close to 0 (NCP), results in a better conditioned posterior than setting it close to 1 (CP), and conversely for strong data ($q = 100$). More interestingly, in intermediate cases ($q = 1$) the optimal value for λ is truly between 0 and 1, yielding a modest but real improvement over the extreme points.

Optimisation. For a general model with latent variables $\mathbf{z}$ and data $\mathbf{x}$, we aim to choose the parameterisation $\boldsymbol{\lambda}$ under which the posterior $p(\tilde{\mathbf{z}} \mid \mathbf{x}; \boldsymbol{\lambda})$ is "most like" an independent normal distribution. A natural objective to minimise is $\mathrm{KL}(q(\tilde{\mathbf{z}}; \boldsymbol{\theta}) \mid\mid p(\tilde{\mathbf{z}} \mid \mathbf{x}; \boldsymbol{\lambda}))$, where $q(\tilde{\mathbf{z}}; \boldsymbol{\theta}) = \mathcal{N}(\tilde{\mathbf{z}} \mid \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}))$ is an independent normal model with *variational parameters* $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$. Minimising this divergence corresponds to maximising a variational lower bound, the ELBO (Bishop, 2006):

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbb{E}_{q(\tilde{\mathbf{z}}; \boldsymbol{\theta})} \left(\log p(\mathbf{x}, \tilde{\mathbf{z}}; \boldsymbol{\lambda}) - \log q(\tilde{\mathbf{z}}; \boldsymbol{\theta}) \right)$$

Note that the auxiliary parameters $\boldsymbol{\lambda}$ are not statistically identifiable: the marginal likelihood $\log p(\mathbf{x}; \boldsymbol{\lambda}) = \log p(\mathbf{x})$ is constant with respect to $\boldsymbol{\lambda}$. However, the computational properties of the reparameterised models differ, and the variational bound will prefer models for which the posterior is close in KL to a diagonal normal. Our key hypothesis (which the results in Figure 6 seem to support) is that diagonal-normal approximability is a good proxy for MCMC sampling efficiency.

To search for a good model reparameterisation, we optimise $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})$ using stochastic gradients to simultaneously fit the

Algorithm 1: Interleaved Hamiltonian Monte Carlo

Arguments: data $\mathbf{x}$; a centred model $M_{cp}(\mathbf{z} \mid \mathbf{x})$
Returns: S samples $\mathbf{z}^{(1)}, \ldots \mathbf{z}^{(S)}$ from $p(\mathbf{z} \mid \mathbf{x})$

$M_{ncp}(\tilde{\mathbf{z}} \mid \mathbf{x}), f = \texttt{make_ncp}(M_{cp}(\mathbf{z} \mid \mathbf{x}))$
$\log p_{cp} = \texttt{make_log_joint}(M_{cp}(\mathbf{z} \mid \mathbf{x}))$
$\log p_{ncp} = \texttt{make_log_joint}(M_{ncp}(\tilde{\mathbf{z}} \mid \mathbf{x}))$

$\mathbf{z}_0 = \texttt{init}()$
for $s \in [1, \ldots, S]$ **do**
  $\mathbf{z}' = \texttt{hmc_step}(\log p_{cp}, \mathbf{z}^{(s-1)})$
  $\mathbf{z}'' = \texttt{hmc_step}(\log p_{ncp}, f^{-1}(\mathbf{z}'))$
  $\mathbf{z}^{(s)} = f(\mathbf{z}'')$
return $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(S)}$

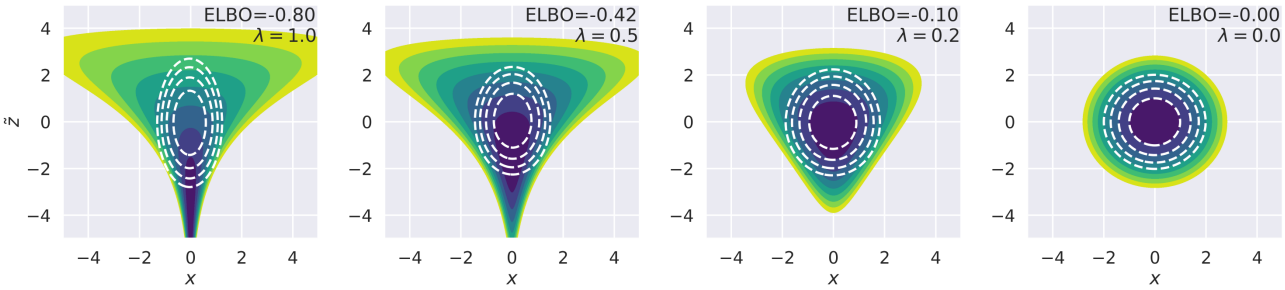
Algorithm 2: Variationally Inferred Parameterisation

Arguments: data $\mathbf{x}$; a centred model $M_{cp}(\mathbf{z} \mid \mathbf{x})$
Returns: S samples $\mathbf{z}^{(1)}, \ldots \mathbf{z}^{(S)}$ from $p(\mathbf{z} \mid \mathbf{x})$

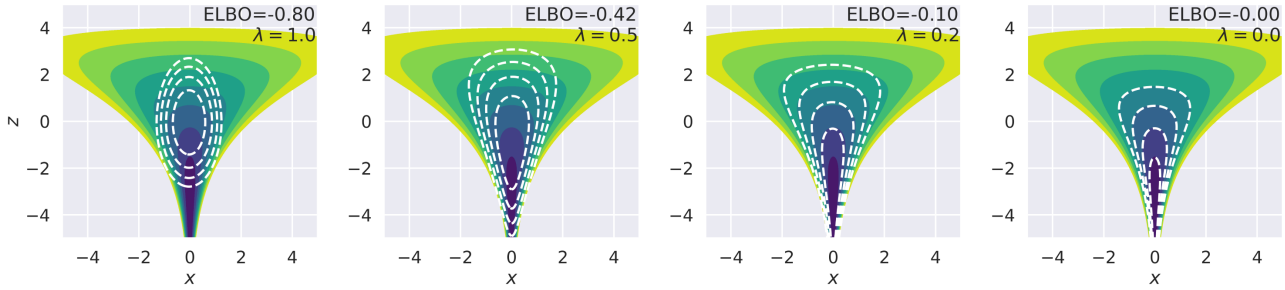
$M_{vip}(\tilde{\mathbf{z}} \mid \mathbf{x}; \boldsymbol{\lambda}), f = \texttt{make_vip}(M_{cp}(\mathbf{z} \mid \mathbf{x}))$
$\log p(\mathbf{x}, \tilde{\mathbf{z}}) = \texttt{make_log_joint}(M_{vip}(\tilde{\mathbf{z}} \mid \mathbf{x}; \boldsymbol{\lambda}))$

$Q(\tilde{\mathbf{z}}; \boldsymbol{\theta}) = \texttt{make_variational}(M_{vip}(\tilde{\mathbf{z}} \mid \mathbf{x}; \boldsymbol{\lambda}))$
$\log q(\tilde{\mathbf{z}}; \boldsymbol{\theta}) = \texttt{make_log_joint}(q(\tilde{\mathbf{z}}; \boldsymbol{\theta}))$

$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbb{E}_q(\log p(\mathbf{x}, \tilde{\mathbf{z}}; \boldsymbol{\lambda})) - \mathbb{E}_q(\log q(\tilde{\mathbf{z}}; \boldsymbol{\theta}))$
$\boldsymbol{\theta}^*, \boldsymbol{\lambda}^* = \operatorname{argmax} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})$
$\log p(\mathbf{x}, \tilde{\mathbf{z}}) = \texttt{make_log_joint}(M_{vip}(\tilde{\mathbf{z}} \mid \mathbf{x}; \boldsymbol{\lambda}^*))$
$\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(S)} = \texttt{hmc}(\log p)$
return $f(\mathbf{z}^{(1)}), \ldots, f(\mathbf{z}^{(S)})$



(a) Different parameterisations $\boldsymbol{\lambda}$ of the funnel, with mean-field normal variational fit $q(\tilde{\mathbf{z}})$(overlayed in white).



(b) Alternative view as implicit variational distributions $q^*_{\boldsymbol{\lambda}}(\mathbf{z})$ (overlayed in white) on the original space.

Figure 3. Neal's funnel: $z \sim N(0, 3)$; $x \sim N(0, e^{z/2})$, with mean-field normal variational fit overlayed.

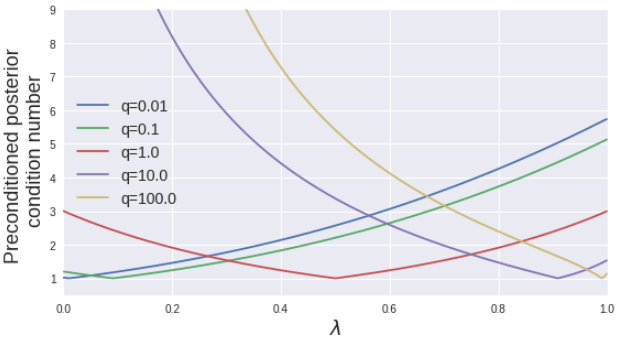


Figure 4. The condition number $\kappa(\lambda)$ for varying $q = N/\sigma^2$ and $\sigma_\mu = 1$ in the simple model from Section 3.

variational distribution q to the posterior p and optimise the shape of that posterior. Figure 3a provides a visual example: an independent normal variational distribution is a poor fit to the pathological geometry of a centred Neal's funnel, but non-centring leads to a well-conditioned posterior, where the variational distribution is a perfect fit. In general settings where the reparameterised model is not exactly Gaussian, sampling-based inference can be used to refine the posterior; we apply VIP as a preprocessing step for HMC (summarised in Algorithm 2). Both the reparameterisation and the construction of the variational model q are implemented as automatic program transformations using Edward2's interceptors.

An alternate interpretation of VIP is that it expands a variational family to a more expressive family capable of rep-

resenting prior dependence. Letting $\mathbf{z} = f_{\boldsymbol{\lambda}}(\tilde{\mathbf{z}})$ represent the partial centring transformation, an independent normal family $q(\tilde{\mathbf{z}})$ on the transformed model corresponds to an implicit posterior $q^*_{\boldsymbol{\lambda}}(\mathbf{z}) = q\left(\tilde{\mathbf{z}} = f^{-1}_{\boldsymbol{\lambda}}(\mathbf{z})\right) |\det J_{f^{-1}_{\boldsymbol{\lambda}}}(\mathbf{z})|$ on the original model variables. Under this interpretation, $\boldsymbol{\lambda}$ are variational parameters that serve to add freedom to the variational family, allowing it to interpolate from independent normal (at $\lambda_i = 1$, Figure 3b left) to a representation that captures the exact prior dependence structure of the model (at $\lambda_i = 0$, Figure 3b right).

6. Experiments

We evaluate the usefulness of our approach as a robust and fully automatic alternative to manual reparameterisation. We compare our methods to HMC ran on fully centred or fully non-centred models, one of which often gives catastrophically bad results. Our results show not only that VIP improves robustness by avoiding catastrophic reparameterisations, but also that it sometimes finds a parameterisation that is better than both the fully centred and fully non-centred alternatives.

6.1. Models and Datasets

We evaluate our proposed approaches by using Hamiltonian Monte Carlo to sample from the posterior of hierarchical Bayesian models on several datasets:

Eight schools (Rubin, 1981): estimating the treatment effects θ_i of a course taught at each of $i = 1 \dots 8$ schools, given test scores y_i and standard errors σ_i:

$$\mu \sim \mathcal{N}(0, 5) \qquad \log \tau \sim \mathcal{N}(0, 5)$$
$$\theta_i \sim \mathcal{N}(\mu, \tau) \qquad y_i \sim \mathcal{N}(\theta_i, \sigma_i)$$

Radon (Gelman & Hill, 2006): hierarchical linear regression, in which the radon level r_i in a home i in county c is modelled as a function of the (unobserved) county-level effect m_c, the county uranium reading u_c, and x_i, the number of floors in the home:

$$\mu, a, b \sim \mathcal{N}(0, 1) \qquad m_c \sim \mathcal{N}(\mu + a u_c, 1)$$
$$\log r_i \sim \mathcal{N}(m_{c[i]} + b x_i, \sigma)$$

German credit (Dua & Graff, 2017): logistic regression; hierarchical prior on coefficient scales:

$$\log \tau_0 \sim \mathcal{N}(0, 10) \qquad \log \tau_i \sim \mathcal{N}(\log \tau_0, 1)$$
$$\beta_i \sim \mathcal{N}(0, \tau_i) \qquad y \sim \text{Bernoulli}(\sigma(\beta X^T))$$

Election '88 (Gelman & Hill, 2006): logistic model of 1988 US presidential election outcomes by county, given demographic covariates $\mathbf{x}_i$ and state-level effects α_s:

$$\beta_d \sim \mathcal{N}(0, 100) \quad \mu \sim \mathcal{N}(0, 100) \quad \log \tau \sim \mathcal{N}(0, 10)$$
$$\alpha_s \sim \mathcal{N}(\mu, \tau) \quad y_i \sim \text{Bernoulli}(\sigma(\alpha_{s[i]} + \beta^T \mathbf{x}_i))$$

Electric Company (Gelman & Hill, 2006): paired causal analysis of the effect of viewing an educational TV show on each of 192 classforms over $G = 4$ grades. The classrooms were divided into $P = 96$ pairs, and one class in each pair was treated ($x_i = 1$) at random:

$$\mu_g \sim \mathcal{N}(0, 1) \quad a_p \sim \mathcal{N}(\mu_{g[p]}, 1) \quad b_g \sim \mathcal{N}(0, 100)$$
$$\log \sigma_g \sim \mathcal{N}(0, 1) \quad y_i \sim \mathcal{N}(a_{p[i]} + b_{g[i]} x_i, \sigma_{g[i]})$$

6.2. Algorithms and Experimental Details

For each model and dataset, we compare our methods, interleaved HMC (iHMC) and VIP-HMC, with baselines of running HMC on either fully centred (CP-HMC) or fully non-centred (NCP-HMC) models. We initialise each HMC chain with samples from an independent Gaussian variational posterior, and use the posterior scales as a diagonal preconditioner; for VIP-HMC this variational optimisation also includes the parameterisation parameters $\boldsymbol{\lambda}$. All variational optimisations were run for the same number of steps, so they were a fixed cost across all methods except iHMC (which depends on preconditioners for both the centred and non-centred transition kernels). The HMC step size and number of leapfrog steps were tuned following the procedures described in Appendix C, which also contains additional details of the experimental setup.

We report the average effective sample size per 1000 gradient evaluations (ESS/∇), with standard errors computed from 200 chains. We use gradient evaluations, rather than wallclock time, as they are the dominant operation in both HMC and VI and are easier to measure reliably; in practice, the wallclock times we observed per gradient evaluation did not differ significantly between methods. This is not surprising, since the (minimal) overhead of interception is incurred only once at graph-building time. This metric is a direct evaluation of the sampler; we do not count the gradient steps taken during the initial variational optimization.

In addition to effective sample size, we also directly examined the convergence of posterior moments for each method. This yielded similar qualitative conclusions to the results we report here; more analysis can be found in Appendix D.

6.3. Results

Figures 5 and 6 show the results of the experiments. In most cases, either the centred or non-centred parameterisation works well, while the other does not. An exception is the German credit dataset, where both CP-HMC and NCP-HMC give a small ESS: 1.2 ± 0.2 or 1.3 ± 0.2 ESS/∇ respectively.

iHMC. Across the datasets in both figures, we see that iHMC is a robust alternative to CP-HMC and NCP-HMC. Its performance is always within a factor of two of the best of CP-HMC and NCP-HMC, and sometimes better. In

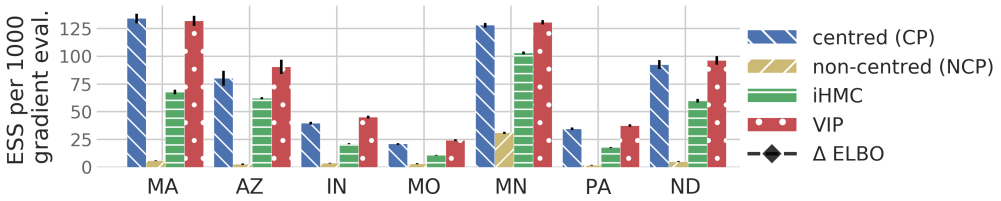


Figure 5. Effective sample size and 95% confidence intervals for the radon model across US states.

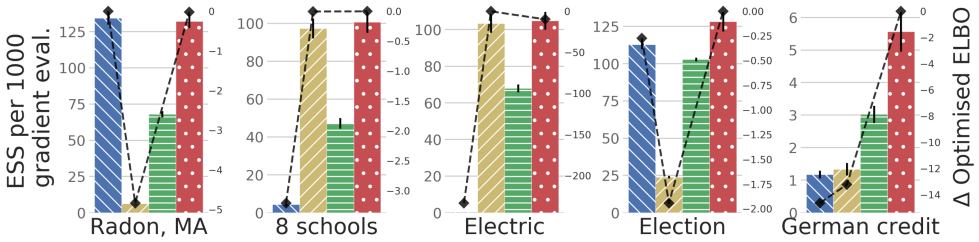


Figure 6. Effective sample size (w/ 95% intervals) and the optimised ELBO across several models.

addition to being robust, iHMC can sometimes navigate the posterior more efficiently than either of CP-HMC and NCP-HMC can: in the case of German credit, it performs better than both (3.0 ± 0.2 ESS$/\nabla$).

VIP. Performance of VIP-HMC is typically as good as the better of CP-HMC and NCP-HMC, and sometimes better. On the German credit dataset, it achieves 5.6 ± 0.6 ESS$/\nabla$, more than three times the rate of CP-HMC and NCP-HMC, and significantly better than iHMC. Figure 6 shows the correspondence between the optimised mean-field ELBO and the effective sampling rate. We see that parameterizations with higher ELBOs tend to yield better samplers, which supports the ELBO as a reasonable predictor of the conditioning of a model.

We show some of the parameterisations that VIP finds in Figure 7. VIP's behaviour appears reasonable: for most datasets we looked at, VIP finds the "correct" global parameterisation: most parameterisation parameters are set to either 0 or 1 (Figure 7, left). In the cases where a global parameterisation is not optimal (e.g. radon MO, radon PA and, most notably, German credit), VIP finds a mixed parameterisation, combining centred, non-centred, and partially centred variables (Figure 7, centre and right). These examples demonstrate the significance of the effect that automatic reparameterisation can have on the quality of inference: manually finding an adequate parameterisation in the German credit case would, at best, require unreasonable amount of hand tuning, while VIP finds such parameterisation automatically.

It is interesting to examine the shape of the posterior landscape under different parameterisations. Figure 8 shows typical marginals of the German credit model. In the centred case, the geometry is funnel-like both in the prior (in grey) and the posterior (in red). In the non-centred case, the prior is an independent Gaussian, but the posteriors still possess significant curvature. The partially centred parameterisations chosen by VIP appear to yield more favourable posterior geometry, where the change in curvature is smaller than that present in the CP and NCP cases.

A practical lesson from our experiments is that while the ELBO appears to correlate with sampler quality, they are not necessarily equally sensitive. A variational model that gives zero mass to half of the posterior is only $\log 2$ away from perfect in the ELBO, but the corresponding sampler may be quite bad. We found it helpful to estimate the ELBO with a relatively large number (tens to hundreds, we used 256) of Monte Carlo samples. As with most variational methods, the VIP optimisation is nonconvex in general, and local optima are also a concern. We occasionally encountered local optima during development, though we found VIP to be generally well-behaved on models for which simpler optimisations are well-behaved. In a practical implementation, one might detect optimization failure by comparing the VIP ELBO to those obtained from fixed parameterizations; for modest-sized models, a deep PPL can often run multiple such optimizations in parallel at minimal cost.

7. Discussion

Our results demonstrate that automated reparameterisation of probabilistic models is practical, and enables inference algorithms that can in some cases find parameterisations

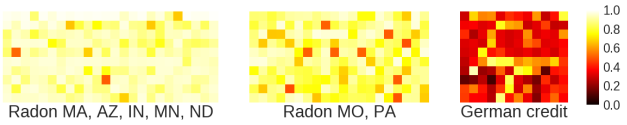


Figure 7. A heat map of VIP parameterisations. Each square represents the obtained using VIP parameterisation parameter λ associated with a different latent variable in the models(s) (e.g. top left corner of German credit corresponds to $\lambda_{\log \tau_1}$). Light regions correspond to CP and dark regions to NCP.

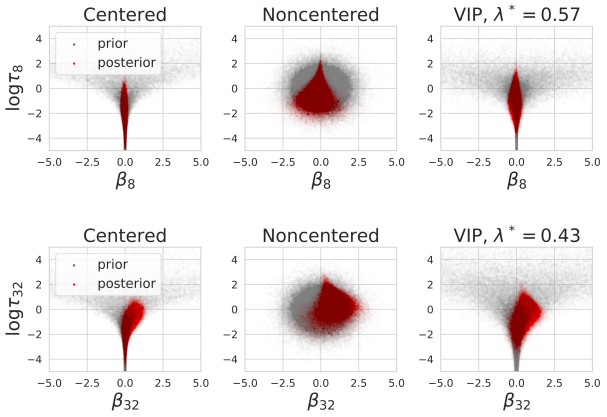


Figure 8. Selected prior and posterior marginals under different parameterisations of the German credit model.

even better than those a human could realistically express. These techniques allow modellers to focus on expressing statistical assumptions, leaving computation to the computer. We view the methods in this paper as exciting proofs of concept, and hope that they will inspire additional work in this space.

Like all variational methods, VIP assumes the posterior can be approximated by a particular functional form; in this case, independent Gaussians 'pulled back' through the noncentring transform. If this family of posteriors does not contain a reasonable approximation of the true posterior, then VIP will not be effective at whitening the posterior geometry. Some cases where this might happen include models where difficult geometry arises from heavy-tailed components (for example, $x \sim Cauchy(0, 1); y \sim Cauchy(x, 1)$), or when the true posterior has structured dependencies that are not well captured by partial centring (for example, many time-series). Such cases can likely be handled by optimising over augmented families of reparameterisations, and designing such families is an interesting topic for future work.

While we focus on reparameterising hierarchical models naturally written in centred form, the inverse transformation—detecting and exploiting implicit hierarchical structure in models expressed as algebraic equations—is an important area of future work. This may be compatible with recent trends exploring the use of symbolic algebra systems in PPL runtimes (Narayanan et al., 2016; Hoffman et al., 2018). We also see promise in automating reparameterisations of heavy-tailed and multivariate distributions, and in designing new inference algorithms to exploit these capabilities.

Acknowledgements

We thank the anonymous reviewers for their useful comments and suggestions. Maria Gorinova was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References

Andrieu, C. and Thoms, J. A tutorial on adaptive MCMC. *Statistics and computing*, 18(4):343–373, 2008.

Betancourt, M. and Girolami, M. Hamiltonian Monte Carlo for hierarchical models. *Current trends in Bayesian methodology with applications*, 79:30, 2015.

Bishop, C. M. *Pattern recognition and machine learning*. Springer, 2006.

Blei, D. M. Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, 1:203–232, 2014.

Cusumano-Towner, M. F., Saad, F. A., Lew, A. K., and Mansinghka, V. K. Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, pp. 221–236, 2019. doi: 10.1145/3314221.3314642.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Gehr, T., Misailovic, S., and Vechev, M. PSI: Exact symbolic inference for probabilistic programs. In *International Conference on Computer Aided Verification*, pp. 62–83. Springer, 2016.

Gehr, T., Steffen, S., and Vechev, M. λPSI: Exact inference for higher-order probabilistic programs. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 883–897, 2020.

Gelman, A. and Hill, J. *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press, 2006.

Gorinova, M. I., Gordon, A. D., and Sutton, C. Probabilistic programming with densities in SlicStan: Efficient, flexible, and deterministic. *Proceedings of the ACM on Programming Languages*, 3(POPL):35, 2019.

Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. NeuTra-lizing bad geometry in Hamiltonian Monte Carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

Hoffman, M. D., Johnson, M., and Tran, D. Autoconj: Recognizing and exploiting conjugacy without a domain-specific language. In *Neural Information Processing Systems*, 2018.

Kastner, G. and Frühwirth-Schnatter, S. Ancillarity-sufficiency interweaving strategy (ASIS) for boosting MCMC estimation of stochastic volatility models. *Computational Statistics & Data Analysis*, 76:408–423, 2014.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Moore, D. and Gorinova, M. I. Effect handling for composable program transformations in Edward2. *International Conference on Probabilistic Programming*, 2018. URL https://arxiv.org/abs/1811.06150.

Narayanan, P., Carette, J., Romano, W., Shan, C., and Zinkov, R. Probabilistic inference by program transformation in Hakaru (system description). In *International Symposium on Functional and Logic Programming - 13th International Symposium, FLOPS 2016, Kochi, Japan, March 4-6, 2016, Proceedings*, pp. 62–79. Springer, 2016. doi: 10.1007/978-3-319-29604-3_5. URL http://dx.doi.org/10.1007/978-3-319-29604-3_5.

Neal, R. M. Slice sampling. *The Annals of Statistics*, 31(3): 705–741, 2003. ISSN 00905364. URL http://www.jstor.org/stable/3448413.

Papaspiliopoulos, O., Roberts, G. O., and Sköld, M. A general framework for the parametrization of hierarchical models. *Statistical Science*, pp. 59–73, 2007.

Parno, M. D. and Marzouk, Y. M. Transport map accelerated Markov chain Monte Carlo. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):645–682, 2018. doi: 10.1137/17M1134640. URL https://doi.org/10.1137/17M1134640.

Plotkin, G. and Power, J. Adequacy for algebraic effects. In Honsell, F. and Miculan, M. (eds.), *Foundations of Software Science and Computation Structures*, pp. 1–24, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-45315-4.

Plotkin, G. and Pretnar, M. Handlers of algebraic effects. In Castagna, G. (ed.), *Programming Languages and Systems*, pp. 80–94, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-00590-9.

Pretnar, M. An introduction to algebraic effects and handlers. Invited tutorial paper. *Electronic Notes in Theoretical Computer Science*, 319:19 – 35, 2015. ISSN 1571-0661. The 31st Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXI).

Rubin, D. B. Estimation in parallel randomized experiments. *Journal of Educational Statistics*, 6(4):377–401, 1981. ISSN 03629791. URL http://www.jstor.org/stable/1164617.

Stan Development Team et al. Stan modelling language users guide and reference manual. *Technical report*, 2016. https://mc-stan.org/docs/2_19/stan-users-guide/.

Tran, D., Hoffman, M. D., Vasudevan, S., Suter, C., Moore, D., Radul, A., Johnson, M., and Saurous, R. A. Simple, distributed, and accelerated probabilistic programming. 2018. URL https://arxiv.org/abs/1811.02091. Advances in Neural Information Processing Systems.

Uber AI Labs. Pyro: A deep probabilistic programming language, 2017. http://pyro.ai/.

Yao, Y., Vehtari, A., Simpson, D., and Gelman, A. Yes, but did it work?: Evaluating variational inference. *arXiv preprint arXiv:1802.02538*, 2018.

Yu, Y. and Meng, X.-L. To center or not to center: That is not the question—an ancillarity–sufficiency interweaving strategy (ASIS) for boosting MCMC efficiency. *Journal of Computational and Graphical Statistics*, 20(3):531–570, 2011.

Zinkov, R. and Shan, C. Composing inference algorithms as program transformations. In Elidan, G., Kersting, K., and Ihler, A. T. (eds.), *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017.