# A. Derivation of the Condition Number of the Posterior for a Simple Model

**Centred Parameterisation**

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

**Non-centred Parameterisation**

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

As the Gaussian distribution is self-conjugate, the posterior distribution (given $\mathbf{x}$) in each case (centred or non-centred) is also a Gaussian distribution, whose shape is entirely specified by a covariance matrix $V$. To quantify the quality of each parameterisation, we investigate the condition number $\kappa$ of the posterior covariance matrix in each case under the best diagonal preconditioner.

We do this in three steps:

1. We derive the covariance matrices $V_{\text{CP}}$ and $V_{\text{NCP}}$, such that $p(\mu, \theta \mid \mathbf{y}) = \mathcal{N}(\mu, \theta \mid \mathbf{m}_{\text{CP}}, V_{\text{CP}})$ and $p(\tilde{\mu}, \theta \mid \mathbf{y}) = \mathcal{N}(\tilde{\mu}, \theta \mid \mathbf{m}_{\text{NCP}}, V_{\text{NCP}})$ (Equation 1 and Equation 2).
2. We find the best diagonal preconditioners $D^*_{\text{CP}}$ and $D^*_{\text{NCP}}$: for $\text{P} = \text{CP}, \text{NCP}$, that is $D^*_{\text{P}} = \arg\min_D(\lambda^{(2)}_{\text{P}}/\lambda^{(1)}_{\text{P}})$, where $\lambda^{(1)}_{\text{P}}$ and $\lambda^{(2)}_{\text{P}}$ are the eigenvalues of $U = D^T V_{\text{P}} D$ (Equation 3 and Equation 4).
3. We compare the condition numbers $\kappa_{cp}(q) = \lambda^{(2)}_{cp}/\lambda^{(1)}_{cp}$ and $\kappa_{ncp}(q) = \lambda^{(2)}_{ncp}/\lambda^{(1)}_{ncp}$, where $\lambda^{(i)}_{(n)cp}$ are the eigenvalues of $U^* = (D^*)^T V D^*$

## A.1. Deriving $V_{\text{CP}}$ and $V_{\text{NCP}}$: Centred Parameterisation

$$\begin{aligned}
p(\mu, \theta \mid \mathbf{y}) &\propto p(\mu, \theta, \mathbf{y}) \\
&\propto \mathcal{N}(\mu \mid \theta, \sigma_\mu)\mathcal{N}(\theta \mid 0, 1) \prod_{n=1}^{N} \mathcal{N}(y_n \mid \mu, \sigma) \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{(\mu-\theta)^2}{\sigma_\mu^2} + \theta^2 + \sum_{n=1}^{N}\frac{(y_n-\mu)^2}{\sigma^2}\right)\right) \\
&\propto \exp\left(-\frac{1}{2}\left(\mu^2\left(\frac{1}{\sigma_\mu^2} + \frac{N}{\sigma^2}\right) + \theta^2\left(\frac{1}{\sigma_\mu^2}+1\right)\right.\right. \\
&\qquad \left.\left. -2\mu\theta\left(\frac{1}{\sigma_\mu^2}\right) + \mu\left(\frac{-2}{\sigma^2}\sum_{n=1}^{N} y_n\right)\right)\right)
\end{aligned}$$

At the same time, for $A = V^{-1}_{\text{NCP}}$, we have:

$$\begin{aligned}
&\mathcal{N}(\mu, \theta \mid \mathbf{m}_{\text{CP}}, V_{\text{CP}}) \\
&\quad \propto \exp\left(-\frac{1}{2}\left(\begin{pmatrix}\mu\\ \theta\end{pmatrix} - \mathbf{m}\right)^T A \left(\begin{pmatrix}\mu\\ \theta\end{pmatrix} - \mathbf{m}\right)\right) \\
&\quad \propto \exp\left(-\frac{1}{2}\left(\mu^2 A_{11}\right.\right. \\
&\qquad\qquad + \theta^2 A_{22} \\
&\qquad\qquad + \mu\theta(2A_{12}) \\
&\qquad\qquad + \mu(-2A_{11}m_1 - 2A_{12}m_2) \\
&\qquad\qquad \left.\left. + \ \mu^2 A_{11}\theta(-2A_{22}m_2 - 2A_{12}m_1)\right)\right)
\end{aligned}$$

Thus, for $q = N/\sigma^2$, we get: $A = \begin{pmatrix} \frac{1}{\sigma_\mu^2} + q & -\frac{1}{\sigma_\mu^2} \\ -\frac{1}{\sigma_\mu^2} & \frac{1}{\sigma_\mu^2} + 1 \end{pmatrix}$

And therefore:

$$V_{\text{CP}} = \frac{1}{\sigma_\mu^2 q + q + 1}\begin{pmatrix} 1+\sigma_\mu^2 & 1 \\ 1 & q\sigma_\mu^2 + 1 \end{pmatrix} \tag{1}$$

## A.2. Deriving $V_{\text{CP}}$ and $V_{\text{NCP}}$: Non-centred Parameterisation

Like in the previous subsection, we have:

$$\begin{aligned}
p(\epsilon, \theta \mid \mathbf{y}) &\propto p(\epsilon, \theta, \mathbf{y}) \\
&\propto \mathcal{N}(\epsilon \mid 0, 1)\mathcal{N}(\theta \mid 0, 1) \prod_{n=1}^{N} \mathcal{N}(y_n \mid \sigma_\mu \epsilon + \theta, \sigma) \\
&\propto \exp\left(-\frac{1}{2}\left((\epsilon^2 + \theta^2 + \sum_{n=1}^{N}\frac{(y_n - \sigma_\mu\epsilon - \theta)^2}{\sigma^2}\right)\right) \\
&\propto \exp\left(-\frac{1}{2}\left(\epsilon^2\left(1 + \frac{N\sigma_\mu^2}{\sigma^2}\right) + \theta^2\left(1 + \frac{N}{\sigma^2}\right)\right.\right. \\
&\qquad + \epsilon\theta\left(\frac{2N\sigma_\mu}{\sigma^2}\right) \\
&\qquad + \epsilon\left(\frac{-2\sigma_\mu \sum y_n}{\sigma^2}\right) \\
&\qquad \left.\left. + \ \theta\left(\frac{-2\sum y_n}{\sigma^2}\right)\right)\right)
\end{aligned}$$

Similarly to before, we derive $A = \begin{pmatrix} \sigma_\mu^2 q + 1 & \sigma_\mu q \\ \sigma_\mu q & q+1 \end{pmatrix}$, and therefore:

$$V_{\text{NCP}} = \frac{1}{\sigma_\mu^2 q + q + 1}\begin{pmatrix} q+1 & -\sigma_\mu q \\ -\sigma_\mu q & \sigma_\mu^2 q + 1 \end{pmatrix} \tag{2}$$

### A.3. The best diagonal preconditioner

Consider a diagonal preconditioner $D = \begin{pmatrix} d & 0 \\ 0 & 1 \end{pmatrix}$. The best diagonal preconditioner $D^*$ of $V$ is such that:

$$D^* = \arg\min_D (\lambda_2/\lambda_1) \text{ where } \lambda_1, \lambda_2 \text{ are the eigenvalues of } U = D^T V D \tag{6}$$

Firstly, in terms of the covariance matrix in the centred case, we have:

$$\begin{aligned} U &= D^T V_{\text{CP}} D \\ &= \begin{pmatrix} d & 0 \\ 0 & 1 \end{pmatrix} \left( \frac{1}{\sigma_\mu^2 q + q + 1} \begin{pmatrix} 1+\sigma_\mu^2 & 1 \\ 1 & q\sigma_\mu^2 + 1 \end{pmatrix} \right) \begin{pmatrix} d & 0 \\ 0 & 1 \end{pmatrix} \\ &= \frac{1}{\sigma_\mu^2 q + q + 1} \begin{pmatrix} (1+\sigma_\mu^2)d^2 & d \\ d & q\sigma_\mu^2 + 1 \end{pmatrix} \end{aligned}$$

The solutions of $det(U - \lambda I) = 0$ are the solutions of:

$$((1+\sigma_\mu^2)d^2 - \lambda(\sigma_\mu^2 q + q + 1))(q\sigma_\mu^2 + 1 - \lambda(\sigma_\mu^2 q + q + 1)) = d^2$$

which, after simplification, becomes:

$$(\sigma_\mu^2 q + q + 1)\lambda^2 - (\sigma_\mu^2 q + 1 + d^2(\sigma_\mu^2 + 1))\lambda + d^2\sigma_\mu^2 = 0$$

We want to find $d$ that minimises $\lambda_2/\lambda_1$. Let $u = d^2$. We are looking for $u$, such that $\frac{\partial}{\partial u}\frac{\lambda_2}{\lambda_1} = 0$, in order to find $d^*_{\text{CP}} = \arg\min_d (\lambda_2/\lambda_1)$. By expanding and simplifying we get:

$$2\frac{\partial}{\partial u}(\sigma_\mu^2 q + 1 + u(\sigma_\mu^2 + 1)) = (\sigma_\mu^2 q + 1 + u(\sigma_\mu^2 + 1))/u$$

And thus:

$$d^*_{\text{CP}} = \sqrt{u} = \sqrt{\frac{\sigma_\mu^2 q + 1}{\sigma_\mu^2 + 1}} \tag{3}$$

We obtain the best diagonal preconditioner $D^*_{\text{NCP}} = \begin{pmatrix} d^*_{\text{NCP}} & 0 \\ 0 & 1 \end{pmatrix}$ in a similar manner, finally getting:

$$d^*_{\text{NCP}} = \sqrt{u} = \sqrt{\frac{\sigma_\mu^2 q + 1}{q + 1}} \tag{4}$$

### A.4. The Condition Numbers $\kappa_{\text{CP}}$ and $\kappa_{\text{NCP}}$

Finally, we substitute $d^*_{\text{CP}}$ and $d^*_{\text{NCP}}$ in the respective eigenvalue equations to derive the condition number in each case:

$$\begin{aligned} \kappa_{\text{CP}} &= \lambda_2^{(\text{CP})}/\lambda_1^{(\text{CP})} \\ &= \frac{\sigma_\mu^2 q + 1 + \sqrt{(\sigma_\mu^2 q+1)^2 - \sigma_\mu^2(\sigma_\mu^2 q + q + 1)(\sigma_\mu^2 q + 1)/(v+1)}}{\sigma_\mu^2 q + 1 - \sqrt{(\sigma_\mu^2 q+1)^2 - \sigma_\mu^2(\sigma_\mu^2 q + q + 1)(\sigma_\mu^2 q + 1)/(v+1)}} \end{aligned} \tag{5}$$

$$\begin{aligned} \kappa_{\text{NCP}} &= \lambda_2^{(\text{NCP})}/\lambda_1^{(\text{NCP})} \\ &= \frac{\sigma_\mu^2 q + 1 + \sqrt{(\sigma_\mu^2 q+1)^2 - \sigma_\mu^2(\sigma_\mu^2 q + q + 1)(\sigma_\mu^2 q + 1)/(q+1)}}{\sigma_\mu^2 q + 1 - \sqrt{(\sigma_\mu^2 q+1)^2 - \sigma_\mu^2(\sigma_\mu^2 q + q + 1)(\sigma_\mu^2 q + 1)/(q+1)}} \end{aligned}$$

## B. Interceptors

Interceptors can be used as a powerful abstractions in a probabilistic programming systems, as discussed previously by Moore & Gorinova (2018), and shown by both Pyro and Edward2. In particular, we can use interceptors to automatically reparameterise a model, as well as to specify variational families. In this section, we show Edward2 pseudo-code for the interceptors used to implement iHMC and VIP-HMC.

### B.1. Make log joint

The following code is an outline of Edward2's impllementation of a function that evaluates the log density $\log p(\mathbf{x})$ at some given $\mathbf{x}$:

```
def make_log_joint_fn(model):
 def log_joint_fn(**kwargs):
  log_prob = 0

  def log_prob_interceptor(
            rv_constructor, **rv_kwargs):
   # Overrides a random variable's value
   # and accumulates its log prob.
   rv_name = rv_kwargs.get("name")
   rv_kwargs["value"]=kwargs.get(rv_name)

   rv = rv_constructor(**rv_kwargs)
   log_prob = log_prob + \
      rv.distribution.log_prob(rv.value)
   return rv

  with ed.interception(
           log_prob_interceptor): model()

  return log_prob
return log_joint_fn
```

By executing the `model` function in the context of `lprob_interceptor`, we override each sample statement (a call to a random variable constructor `rv_constructor`), to generate a variable that takes on the value provided in the arguments of `log_joint_fn`. As a side effect, we also accumulate the result of evaluating each variable's prior density at the provided value, which, by the chain rule, gives us the log joint density.

### B.2. Non-centred Parameterisation Interceptor

By intercepting every construction of a normal variable (or, more generally, of location-scale family variables), we can

create a standard normal variable instead, and scale and shift appropriately.

```
def ncp_interceptor(rv_constr,
                    **rv_kwargs):
 # Assumes rv_constr is in the
 # location-scale family
 name = rv_kwargs["name"] + "_std"
 rv_std = \
  ed.interceptable[6](rv_constr)(
                        loc=0, scale=1)
 return rv_kwargs["loc"] + \
        rv_kwargs["scale"] * rv_std
```

Running a model that declares the random variables $\boldsymbol{\theta}$ in the context of `ncp_interceptor` will declare a new set of standard normal random variables $\boldsymbol{\theta}^{(\text{std})}$. Nesting this in the context of the `log_prob_interceptor` from **??** will then evaluate the log joint density $\log p(\boldsymbol{\theta}^{(\text{std})})$.

For example, going back to Neal's funnel, running

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

corresponds to evaluating $\log p(z, x) = \log \mathcal{N}(z \mid 0, 3) + \log \mathcal{N}(x \mid 0, e^{z/2})$, while running

```
with ed.interception(
          log_prob_interceptor):
 with ed.interception(
          ncp_interceptor): neals_funnel()
```

corresponds to evaluating $\log p(z^{(\text{std})}, x^{(\text{std})}) = \log \mathcal{N}(z^{(\text{std})} \mid 0, 1) + \log \mathcal{N}(x^{(\text{std})} \mid 0, 1)$.

### B.3. VIP Interceptor

The VIP interceptor is similar to the NCP interceptor. The notable difference is that it creates new learnable Tensorflow variables, which correspond to the parameterisation parameters $\boldsymbol{\lambda}$:

```
def vip_interceptor(rv_constr,
                    **rv_kwargs):
 name = rv_kwargs["name"] + "_vip"
 rv_loc = rv_kwargs["loc"]
 rv_scale = rv_kwargs["scale"]

 a = tf.nn.sigmoid(tf.get_variable(
   name + "_a_unconstrained",
   initializer=tf.zeros_like(rv_loc))

 rv_vip = ed.interceptable(rv_constr)(
          loc=a*rv_loc, scale=rv_scale**a)
 return rv_loc + \
   rv_scale**(1-a) * (rv_vip - a*rv_loc)
```

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

### B.4. Mean-field Variational Model Interceptor

Finally, we show a mean-field variational familiy interceptor, which we use both to tune the step sizes for HMC (see Appendix C), and to make use of VIP automatically. The `mfvi_interceptor` simply substitutes each sample statement with sampling from a normal distribution with parameters specified by some fresh variational parameters $\mu$ and $\sigma$:

```
def vip_interceptor(rv_constructor,
                    **rv_kwargs):
 name = rv_kwargs["name"] + "_q"
 mu = tf.get_variable(name + "_mu")
 sigma = tf.nn.softmax(
          tf.get_variable(
                    name + "_sigma"))

 rv_q = ed.interceptable(ed.Normal)(
      loc=mu, scale=sigma, name=name)
 return rv_q
```

## C. Details of the Experiments

**Algorithms.**

- CP-HMC: HMC run on a fully centred model.
- NCP-HMC: HMC run on a fully non-centred model.
- iHMC: interleaved HMC.
- VIP-HMC: HMC run on the a model reparameterised as given by VIP.

Each run consists of VI pre-processing and HMC inference.

**Variational Inference Pre-processing.** We use automatic differentiation to compute stochastic gradients of the ELBO with respect to $\boldsymbol{\lambda}, \boldsymbol{\theta}$ and perform the optimisation using Adam (Kingma & Ba, 2014). We implement the constraint $\lambda_i \in [0, 1]$ using a sigmoid transformation; $\lambda_i = 1/\left(1 + \exp(-\tilde{\lambda}_i)\right)$ for $\tilde{\lambda}_i \in \mathbb{R}$.

Prior to running HMC, we also run VI to approximate per-variable initial step sizes (equivalently, a diagonal preconditioning matrix), and to initialise the chains. For each of CP-HMC and NCP-HMC this is just mean-field VI, and for VIP-HMC the VI procedure is VIP.

Each VI method is run for 3000 optimisation steps, and the ELBO is approximated using 256 Monte Carlo samples. We use the Adam optimiser with initial learning rate $\alpha \in [0.02, 0.05, 0.1, 0.2, 0.4]$, decayed to $\alpha/5$ after 1000 steps and $\alpha/20$ after 2000 steps, and returned the result with the highest ELBO.

**Hamiltonian Monte Carlo Inference.** In each case we run 200 chains for a warm-up period of 2000 steps, followed by 10000 steps each, and report the average effective sample size (ESS) per 1000 gradient evaluations (ESS/$\nabla$). Since ESS is naturally estimated from scalar traces, we first estimate per-variable effective sample sizes for each model variable, and take the overall ESS to be the minimum across all variables.

The HMC step size $s_t$ was adapted to target an acceptance probability of 0.75, following a simple update rule

$$\log s_{t+1} = \log s_t + 0.02 \cdot (\mathbb{I}[\alpha_t - 0.75] - \mathbb{I}[0.75 - \alpha_t])$$

where $\alpha_t$ is the acceptance probability of the proposed state at step $t$ (Andrieu & Thoms, 2008). The adaptation runs during the first 1500 steps of the warm-up period, after which we allow the chain to mix towards a stationary distribution.

The number of leapfrog steps is chosen using 'oracle' tuning: each sampler is run with logarithmically increasing number of leapfrog steps in $\{1, 2, 4, \ldots, 128\}$, and we report the result that maximises ESS/$\nabla$. This is intended to decouple the problem of tuning the number of leapfrog steps from the issues of parameterisation consider in this paper, and ensure that each method is reasonably tuned. For iHMC, we tune a single number of leapfrog steps that is shared across both the CP and NCP substeps.

## D. Additional Analysis

In addition to the estimated effective sample sizes, we directly examined posterior moments estimated from each method. Theory implies that the estimated posterior mean and standard deviation should converge to their true values at a rate of $O(\sqrt{N})$, where $N$ is the number of effective samples, so we would expect these results to be broadly consistent with the estimated effective samples sizes in § 6.

For each model, we computed a 'gold standard' estimated posterior mean and standard deviation. We first used each method to estimate empirical means and standard deviations for the latent variables, using the full set of $200 \times 10000$ samples produced across all chains. We then took the median of each statistic across the four methods, i.e., the mean of the two central values, for our final estimate. This is robust to the case where one of the four methods totally fails to mix— generally because a fully centred or fully noncentred parameterisation is not appropriate—as long as the other methods produce reasonable samples. By inspection, at least three of the four methods agreed closely in all cases, which provides some comfort that our gold standard estimates are reasonable.

Table 1 shows normalized expected error in the posterior statistics estimated by a *single chain* of each method, as a function of the number of gradient steps taken. For each latent variable $z_i$, we compute the expected absolute error in the mean, using the running mean $\hat{\mu}_{i,k}^{(:t)}$ estimated from the first $t$ gradient steps of the $k$th chain, and the gold standard mean $\mu_i$ computed as above, to be

$$r_i^{(:t)} = \frac{1}{K} \sum_k \left| \hat{\mu}_{i,k}^{(:t)} - \mu_i, \right|$$

where $K$ is the number of chains. We analogously compute expected absolute error in standard deviation by

$$s_i^{(:t)} = \frac{1}{K} \sum_k \left| \hat{\sigma}_{i,k}^{(:t)} - \sigma_i \right|.$$

Each model has many latent variables, whose posteriors have different scales. To summarise inference across all variables in a model, we report the mean of the errors normalized by the standard deviation (which we treat as a reasonable representative of the posterior scale for each variable) across all $N$ latent variables,

$$\bar{r}^{(:t)} = \frac{1}{N} \sum_i r_i^{(:t)} / \sigma_i \qquad \bar{s}^{(:t)} = \frac{1}{N} \sum_i s_i^{(:t)} / \sigma_i.$$

These quantities are plotted in Table 1. Note that the x axis is the number of gradient steps taken; this may correspond to different numbers of actual samples from each method, depending on the number of leapfrog steps used. As discussed in § 6, the number of gradient steps is the appropriate metric here, as it's a reasonable proxy for wallclock time.

The relative performance of the methods generally corresponds to and supports the effective sample size estimates reported in § 6. VIP converges notably faster than other methods in the German credit model, and otherwise about as quickly as the better of CP and NCP, each of which sometimes fails quite badly on its own. Interleaved HMC is generally between CP and NCP, with the exception of the estimated standard deviations on the electric company dataset, where it notably beats out all of the other methods.

We also provide autocorrelations for each method, plotting autocorrelations for each latent variable across three chains along with the mean autocorrelation (plotted in bold). Given an autocorrelation sequence $r_1, \ldots, r_{N-1}$ on $N$ samples, the effective sample size is defined as

$$ESS(N) = \frac{N}{1 + 2\sum_{i=1}^{N-1} \frac{N-i}{N} r_i};$$

these are the values reported in § 6.

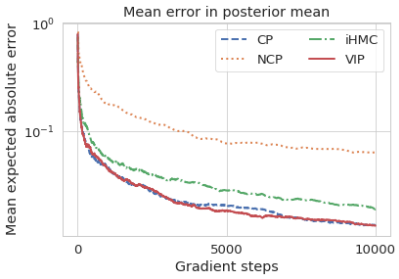
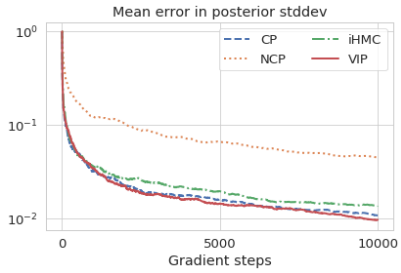
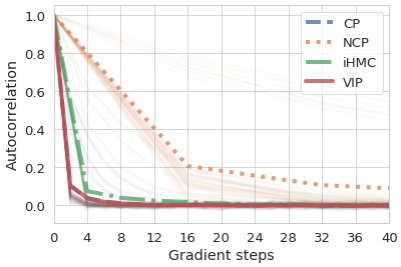
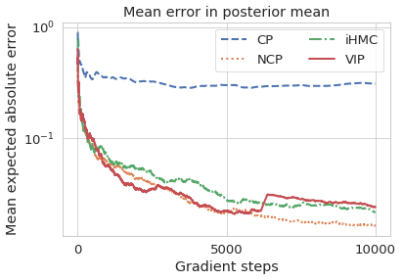
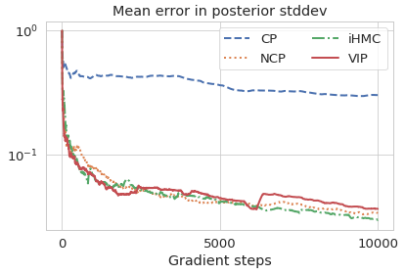
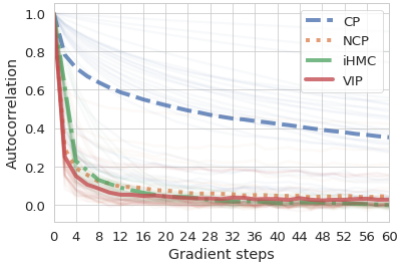
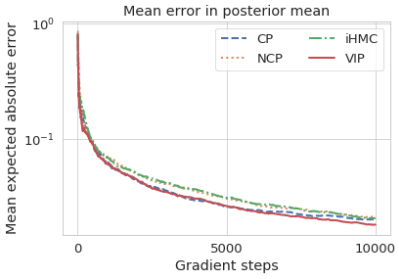
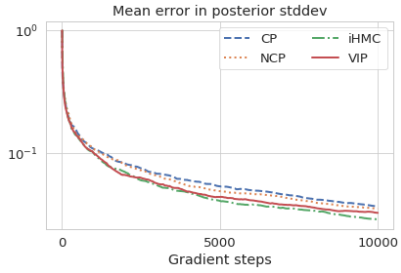
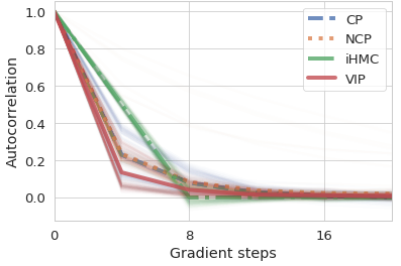
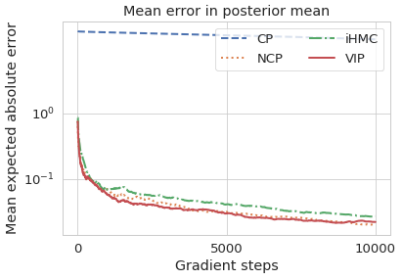
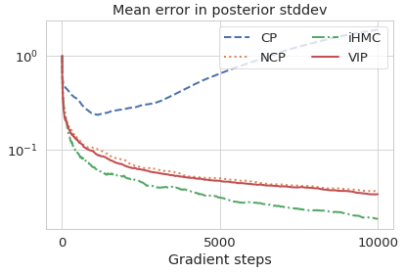
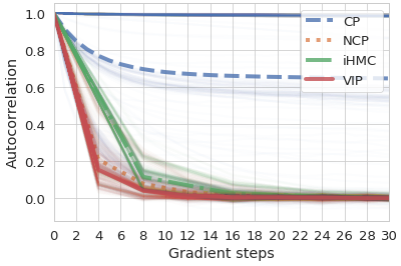
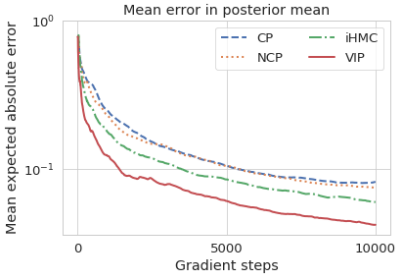
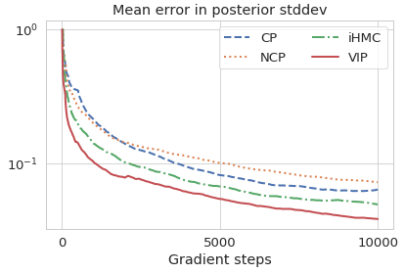
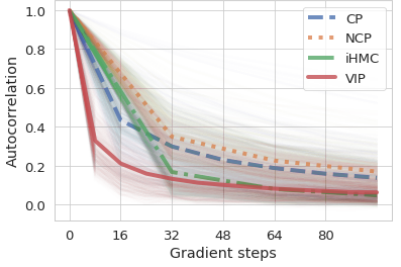
| Error in mean ($\bar{r}^{(:t)}$) | Error in stddev ($\bar{s}^{(:t)}$) | Autocorrelations |
|---|---|---|
| Radon, MA | | |
|  |  |  |
| Eight schools | | |
|  |  |  |
| Election | | |
|  |  |  |
| Electric | | |
|  |  |  |
| German credit | | |
|  |  |  |

*Table 1.* Additional inference diagnostics.