HMC vs Gibbs Sampling

February 3, 2026

1 Hamiltonian Monte Carlo (HMC)

1.1 Overview

Figure 1 shows the geometry of one HMC iteration in parameter space. We start at our current position $\theta^{(t)}$ (the blue point), introduce momentum p (the blue arrow), and then simulate Hamiltonian dynamics guided by the gradient $\nabla \log p(\theta|y)$ (the green arrow). This creates a curved trajectory (the blue path) that proposes a new point θ^{proposal} (the orange point). Finally, we accept or reject this proposal using a Metropolis step.

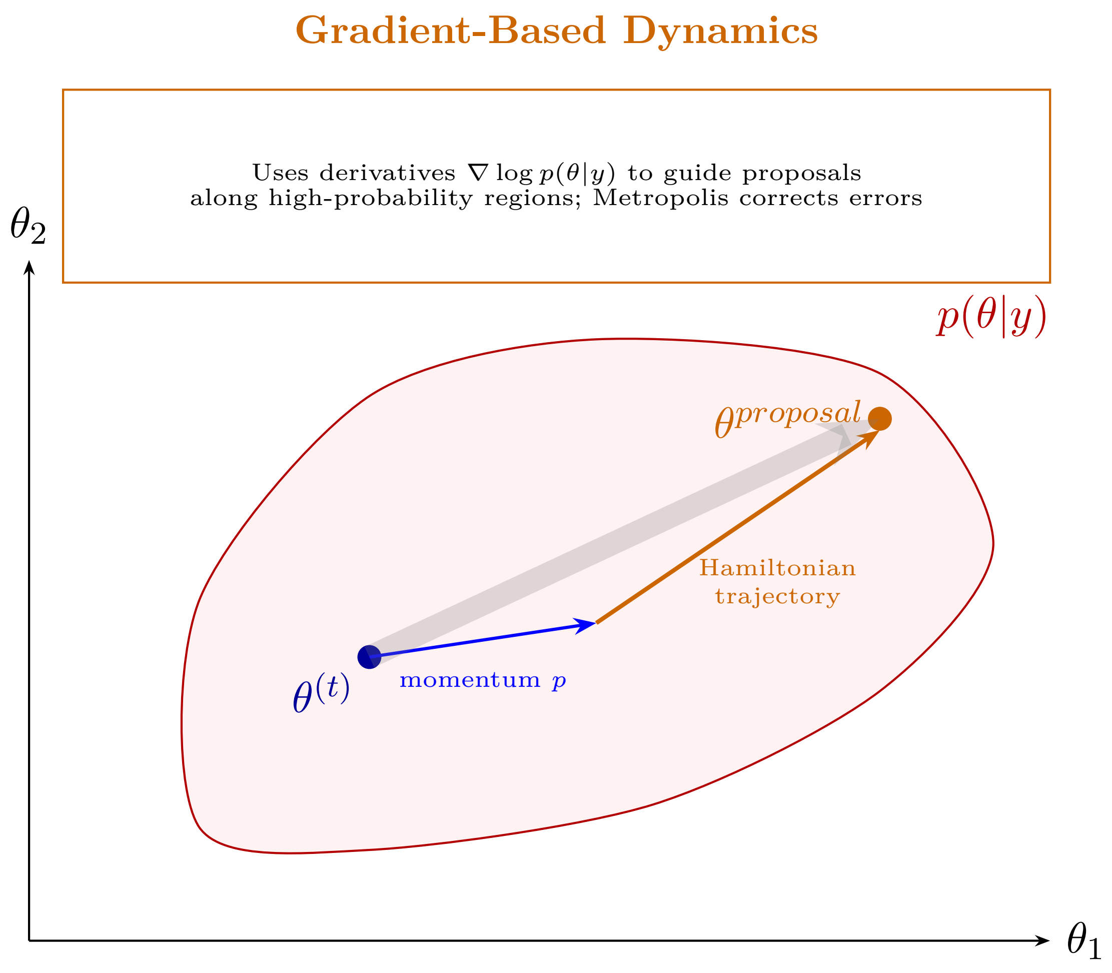

Figure 1: HMC visual summary: current position, momentum, gradient direction, and Hamiltonian trajectory leading to a proposal.

The key idea behind HMC is to use auxiliary variablesextra variables not part of the original model that help the algorithm work more efficiently. Specifically, HMC adds momentum variables p and then simulates Hamiltonian dynamics, treating parameters as positions and momentum as velocities according to physics-inspired equations. This simulation uses the derivative of the log posterior $\nabla \log p(\theta|y)$, which tells us how the posterior probability changes as we move through parameter space. We compute the derivative with respect to each parameter (the partial derivative $\partial \log p(\theta|y)/\partial\theta_j$) to guide the trajectory towards high-probability regions. Finally, a Metropolis accept–reject step corrects for numerical errors in the simulation, ensuring the sampler converges to the correct posterior distribution (a property called detailed balance).

1.2 Walking Through One HMC Iteration

We now walk through one iteration of HMC step by step, showing the notation, the variables that exist at each stage, and how they connect to the diagram in Figure 1.

Step 0: Setup and notation

Before we begin sampling, we define our model and prior. For example:

$$p(\theta) \quad \text{with} \quad \beta \sim \mathcal{N}(0, 100).$$

Our target is the posterior distribution:

$$p(\theta|y) \propto p(y|\theta)p(\theta).$$

We start at the previous sample $\theta^{(n-1)}$. This is the blue point $\theta^{(t)}$ in Figure 1. At this stage, we have just one vector:

- $\theta^{(n-1)}$ (56 parameter values)

Step 1: Sample momentum

HMC introduces auxiliary momentum variables p drawn independently from:

$$p_0 \sim \mathcal{N}(0, I).$$

This is the blue arrow labelled "momentum p" in Figure 1. Setting $\theta_0 = \theta^{(n-1)}$, we now have two vectors:

- $\theta_0 = \theta^{(n-1)}$ (56 parameter values)
- p_0 (56 momentum values)

In code, this looks like:

```
theta <- theta_start
p     <- rnorm(length(theta), mean = 0, sd = 1)
```

Step 2: Leapfrog trajectory

Starting from (θ_0, p_0), we apply L leapfrog updates. Each update alternates between a half-step momentum update using the gradient $\nabla \log p(\theta|y)$ (the green arrow in Figure 1), a full-step position update, and another half-step momentum update. This creates the sequence:

$$(\theta_0, p_0) \to (\theta_1, p_1) \to \cdots \to (\theta_L, p_L).$$

The curved blue path in Figure 1 represents this Hamiltonian trajectory. During the trajectory, we have intermediate states (θ_k, p_k) for $k = 0, \ldots, L$, but we only keep the endpoint. After the trajectory completes, we have four vectors:

- $\theta^{(n-1)} = \theta_0$ (56 starting parameter values)
- p_0 (56 starting momentum values)
- θ_L (56 new parameter values after L steps)
- p_L (56 updated momentum values after L steps)

In code:

```
for (k in 1:L) {
    grad  <- grad_log_posterior(theta, y)
    p     <- p + 0.5 * eps * grad
    theta <- theta + eps * p
    grad  <- grad_log_posterior(theta, y)
    p     <- p + 0.5 * eps * grad
}
```

Step 3: Proposal

We set the proposal to be the endpoint of the trajectory:

$$\begin{aligned} \theta^{\text{proposal}} &= \theta_L \\ p^{\text{proposal}} &= p_L \end{aligned}$$

This is the orange point θ^{proposal} in Figure 1. We still have the same four vectors as before, just relabelled:

- $\theta^{(n-1)}$ (56 starting values)
- p_0 (56 starting momentum values)
- θ^{proposal} (56 proposed new values)
- p^{proposal} (56 final momentum values)

In code:

```
theta_proposal <- theta
```

Step 4: Accept or reject

We compute the log acceptance probability:

$$\log \alpha = \log p(\theta^{\text{proposal}}|y) - \log p(\theta^{(n-1)}|y).$$

Then we accept the proposal with probability $\min(1, e^{\log \alpha})$:

$$\theta^{(n)} = \begin{cases} \theta^{\text{proposal}} & \text{with probability } \min(1, e^{\log \alpha}) \\ \theta^{(n-1)} & \text{otherwise} \end{cases}$$

This Metropolis step corrects for numerical errors from the leapfrog approximation. After this step, the momentum values are discarded (they were auxiliary), and we have just one vector:

- $\theta^{(n)}$ (56 parameter values for the next iteration)

In code:

```
log_post_prop <- log_posterior(theta_proposal, y)
log_post_curr <- log_posterior(theta_start, y)
log_alpha     <- log_post_prop - log_post_curr

if (log(runif(1)) < log_alpha) {
    theta_next <- theta_proposal
} else {
    theta_next <- theta_start
}
```

Step 5: Repeat

We return to Step 1 with $\theta^{(n-1)} = \theta^{(n)}$ and continue sampling.

2 Blocked Gibbs Sampling

2.1 Overview

Figure 2 shows the geometry of one blocked Gibbs iteration. We start at our current position $\theta^{(t)}$ (the blue point), which contains all parameters $(\beta, u, \tau_e, \tau_u)$. Unlike HMC, which uses gradients and momentum, Gibbs sampling updates parameters sequentially by sampling from full conditional distributions. Each parameter (or block of parameters) is updated by drawing from its conditional distribution given all other parameters and the data. The three coloured arrows show the three moves: sampling τ_u (blue vertical arrow), sampling (β, u) jointly (green horizontal arrow), and sampling τ_e (purple vertical arrow), arriving at $\theta^{(t+1)}$ (the purple point).

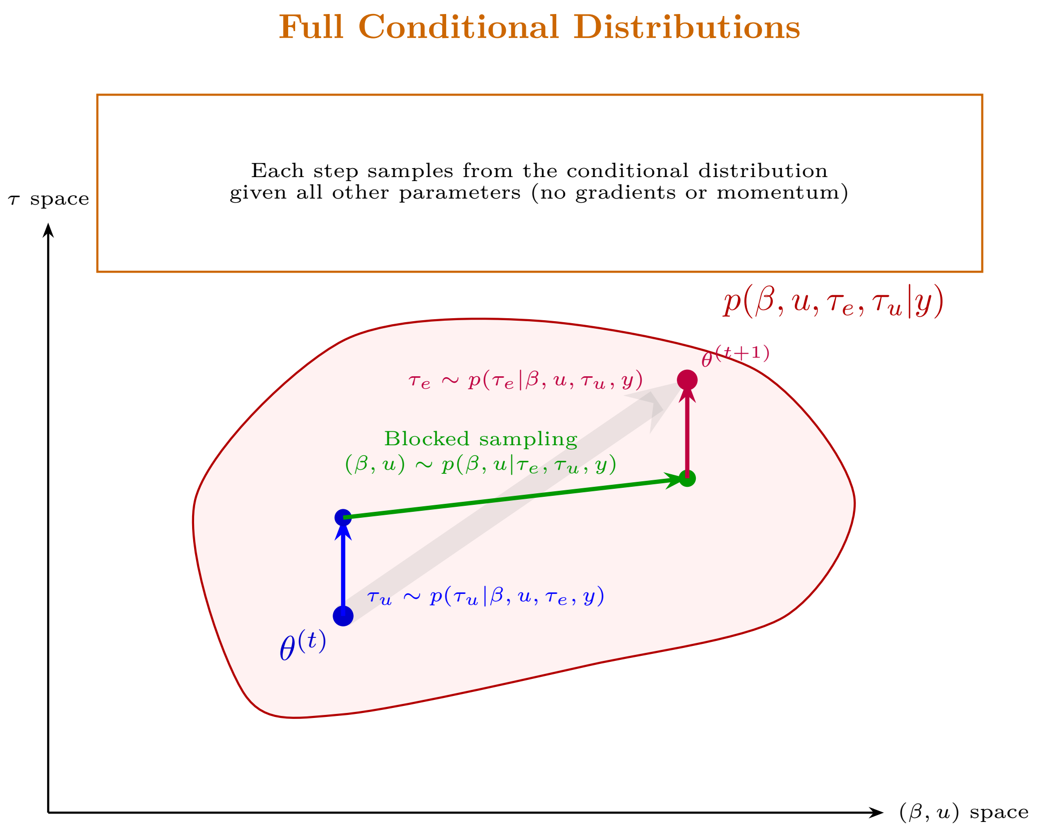
Full Conditional Distributions
Each step samples from the conditional distribution
given all other parameters (no gradients or momentum)
τ space
$p(\beta, u, \tau_e, \tau_u|y)$
$\theta^{(t+1)}$
$\tau_e \sim p(\tau_e|\beta, u, \tau_u, y)$
Blocked sampling
$(\beta, u) \sim p(\beta, u|\tau_e, \tau_u, y)$
$\tau_u \sim p(\tau_u|\beta, u, \tau_e, y)$
$\theta^{(t)}$
(β, u) space

Figure 2: Blocked Gibbs sampling: three sequential updates via full conditional distributions.

The key idea behind Gibbs sampling is to use full conditional distributionsthe probability distribution of one parameter (or block of parameters) conditional on all others and the data. These distributions often have closed-form expressions, allowing direct sampling without gradients, momentum variables, or acceptance steps. For variance components τ_e and τ_u, the full conditionals are typically inverse-gamma distributions. For the regression parameters (β, u) given the variances, the full conditional is a multivariate normal. By cycling through these conditionals, the sampler explores the joint posterior without ever computing derivatives.

2.2 Walking Through One Gibbs Iteration

We now walk through one iteration of blocked Gibbs sampling step by step, showing the notation, the variables that exist at each stage, and how they connect to the diagram in Figure 2.

Step 0: Setup and notation

Our hierarchical model is:

$$\begin{aligned} &y = X\beta + Zu + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \tau_e^{-1} I) \\ &u \sim \mathcal{N}(0, \tau_u^{-1} K) \end{aligned}$$

where $\theta = (\beta, u, \tau_e, \tau_u)$ contains all parameters. Our target is the posterior:

$$p(\beta, u, \tau_e, \tau_u | y) \propto p(y|\beta, u, \tau_e) p(u|\tau_u) p(\beta) p(\tau_e) p(\tau_u).$$

Note that each hyperparameter appears twice: τ_e appears in the likelihood $p(y|\beta, u, \tau_e)$ and in its own prior $p(\tau_e)$; similarly, τ_u appears in the conditional prior for u and in its own prior. This structure is characteristic of Bayesian hierarchical models, where hyperparameters govern lower-level distributions whilst themselves being treated as random variables with their own priors Gelman and Hill (2006); Gelman (2006).

We start at the previous sample $\theta^{(t)} = (\beta^{(t)}, u^{(t)}, \tau_e^{(t)}, \tau_u^{(t)})$. This is the blue point in Figure 2. At this stage, we have:

- $\beta^{(t)}$ (4 values)
- $u^{(t)}$ (50 values)
- $\tau_e^{(t)}$ (1 value)
- $\tau_u^{(t)}$ (1 value)

Total: 56 parameter values in $\theta^{(t)}$.

Step 1: Sample τ_u

We sample the random effect variance component from its full conditional:

$$\tau_u^{(t+1)} \sim p(\tau_u | \beta^{(t)}, u^{(t)}, \tau_e^{(t)}, y).$$

For our model, this is an inverse-gamma distribution:

$$\tau_u^{(t+1)} \sim \text{InvGamma}\left(a_u + \frac{n_u}{2},\, b_u + \frac{1}{2} u^{(t)T} K^{-1} u^{(t)}\right).$$

This is the blue vertical arrow in Figure 2, moving in τ space whilst holding (β, u) fixed. After this step, we have:

- $\beta^{(t)}$ (4 values, unchanged)
- $u^{(t)}$ (50 values, unchanged)
- $\tau_e^{(t)}$ (1 value, unchanged)
- $\tau_u^{(t+1)}$ (1 new value)

In code:

```
shape_u <- a_u + n_u / 2
rate_u  <- b_u + 0.5 * t(u) %*% solve(K) %*% u
tau_u   <- rgamma(1, shape = shape_u, rate = rate_u)
```

Step 2: Sample (β, u) jointly

We sample the regression parameters and random effects jointly from their full conditional:

$$(\beta^{(t+1)}, u^{(t+1)}) \sim p(\beta, u | \tau_e^{(t)}, \tau_u^{(t+1)}, y)$$

For our model, this is a multivariate normal distribution. The joint update (blocked sampling) is crucial for good mixing when β and u are correlated. This is the green horizontal arrow in Figure 2, moving in (β, u) space whilst holding (τ_e, τ_u) fixed. After this step, we have:

- $\beta^{(t+1)}$ (4 new values)
- $u^{(t+1)}$ (50 new values)
- $\tau_e^{(t)}$ (1 value, unchanged)
- $\tau_u^{(t+1)}$ (1 value from Step 1)

In code:

```
# Precision matrix for joint (beta, u)
Q       <- tau_e * t(cbind(X, Z)) %*% cbind(X, Z)
Q[5:54, 5:54] <- Q[5:54, 5:54] + tau_u * solve(K)

# Mean vector
mu      <- solve(Q) %*% (tau_e * t(cbind(X, Z)) %*% y)

# Sample jointly
theta   <- mvrnorm(1, mu = mu, Sigma = solve(Q))
beta    <- theta[1:4]
u       <- theta[5:54]
```

Step 3: Sample τ_e

We sample the error variance component from its full conditional:

$$\tau_e^{(t+1)} \sim p(\tau_e | \beta^{(t+1)}, u^{(t+1)}, \tau_u^{(t+1)}, y).$$

For our model, this is an inverse-gamma distribution:

$$\tau_e^{(t+1)} \sim \text{InvGamma}\left(a_e + \frac{n}{2},\, b_e + \frac{1}{2}(y - X\beta^{(t+1)} - Zu^{(t+1)})^T(y - X\beta^{(t+1)} - Zu^{(t+1)})\right).$$

This is the purple vertical arrow in Figure 2, moving in τ space whilst holding (β, u, τ_u) fixed. After this step, we arrive at $\theta^{(t+1)}$ (the purple point), and we have:

- $\beta^{(t+1)}$ (4 values from Step 2)
- $u^{(t+1)}$ (50 values from Step 2)
- $\tau_e^{(t+1)}$ (1 new value)
- $\tau_u^{(t+1)}$ (1 value from Step 1)

Total: 56 parameter values in $\theta^{(t+1)}$.

In code:

```
resid    <- y - X %*% beta - Z %*% u
shape_e  <- a_e + n / 2
rate_e   <- b_e + 0.5 * sum(resid^2)
tau_e    <- rgamma(1, shape = shape_e, rate = rate_e)
```

Step 4: Repeat

We return to Step 1 with $\theta^{(t)} = \theta^{(t+1)}$ and continue sampling.

2.3 Key Differences from HMC

- No gradients: Gibbs samples directly from full conditionals; HMC uses $\nabla \log p(\theta|y)$ to guide trajectories
- No auxiliary variables: Gibbs updates parameters in place; HMC introduces momentum p
- No accept/reject: Gibbs always accepts (samples are exact from conditionals); HMC uses Metropolis correction
- Sequential updates: Gibbs cycles through parameter blocks; HMC updates all parameters jointly via the trajectory
- Closed-form sampling: Gibbs relies on tractable conditionals (inverse-gamma, multivariate normal); HMC works even when conditionals are intractable

Both methods explore the posterior $p(\theta|y)$, but via completely different mechanisms. Gibbs is simpler when conditionals are available, whilst HMC is more robust to correlations and scales better to high dimensions.

References

Andrew Gelman. Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, 1(3):515–534, 2006.

Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, Cambridge, 2006.