Assignment 1: [Adaptive Learned Controller Step-size HMC (ALCS-HMC)]

Sungjoon Park
Boston University
spark618@bu.edu

Abstract

I propose **ALCS-HMC** (Adaptive Learned Controller Step-size HMC), which replaces HMC's fixed step size with $\varepsilon_{\text{eff}}(q) = \varepsilon_{\text{base}} \cdot \alpha(\log \|\nabla E(q)\|; \theta)$, where α is a small MLP trained online by back-propagating through the differentiable leapfrog trajectory. The controller learns to shrink steps on high-curvature ridges and in funnel necks, and to expand them in flat regions—adapting to geometry that defeats both random-walk MH and fixed-step HMC. After a warm-up phase the MLP is frozen, giving a consistent sampling policy. Experiments show competitive acceptance rates and improved moment recovery versus HMC baselines on the Rosenbrock and Neal's funnel benchmarks.

1 Introduction

Efficient posterior sampling requires step sizes tuned to local geometry. Random Walk MH uses isotropic proposals that cannot follow curved ridges; HMC uses gradient information but still requires a *fixed* global step size ε. This creates two canonical failure modes. On the **Rosenbrock distribution** ($\log p \propto -0.05(1-x)^2 - (y-x^2)^2$), the high-probability region is a thin curved ridge: a fixed ε large enough to traverse the banana will step off it; a conservative ε keeps the chain locally jittering. On **Neal's funnel** ($v \sim \mathcal{N}(0, 9)$, $x \,|\, v \sim \mathcal{N}(0, e^v)$), the optimal step size spans orders of magnitude as v varies; any fixed choice either diverges in the narrow neck or stalls in the wide mouth.

Both problems share a single root cause: the optimal ε is a *function of position*, and the gradient norm $\|\nabla E(q)\|$ is a cheap, informative proxy for it—large on ridges and in funnel necks, small in flat regions. ALCS-HMC exploits this by learning the mapping $\log \|\nabla E(q)\| \mapsto \alpha$ with a small MLP, giving $\varepsilon_{\text{eff}}(q) = \varepsilon_{\text{base}} \cdot \alpha$.

2 Method

Step-size controller. A two-hidden-layer MLP ($\tanh$ activations, width 16, `softplus` output) maps $\phi(q) = \log(\|\nabla E(q)\| + 10^{-8})$ to a multiplier $\alpha \in [0.10,\ 2.50]$, giving $\varepsilon_{\text{eff}}(q) = \varepsilon_{\text{base}} \cdot \alpha(\phi(q); \theta)$. The output bias is initialised to $\texttt{softplus}^{-1}(1) \approx 0.541$ so $\alpha \approx 1$ before training (standard HMC warm start). The clip bounds ensure $\varepsilon_{\text{eff}} \leq 2.5\, \varepsilon_{\text{base}}$, keeping velocity-Verlet within its stability region ($\varepsilon\sqrt{\lambda_{\max}} < 2$), while preventing complete chain stall from below.

Leapfrog and acceptance. ε_{eff} is evaluated once at q_0 and held fixed for all L leapfrog steps. The proposal (q_L, p_L) is accepted with probability $\min(1, e^{-\Delta H})$, $\Delta H = H(q_L, p_L) - H(q_0, p_0)$.

Detailed balance. Holding $\varepsilon_{\text{eff}}(q_0)$ fixed throughout breaks time-reversal symmetry: the reverse trajectory from q_L would use $\varepsilon_{\text{eff}}(q_L) \neq \varepsilon_{\text{eff}}(q_0)$, so $\min(1, e^{-\Delta H})$ is a *biased* acceptance ratio. An exact fix adds the log-step-size correction $\log \varepsilon_{\text{eff}}(q_L) - \log \varepsilon_{\text{eff}}(q_0)$ to the log-acceptance; an

Preprint.

Algorithm 1 ALCS-HMC

Require: q_0, $\varepsilon_{\text{base}}$, L, T_w, T, lr η
Init θ with output bias $\texttt{softplus}^{-1}(1)$
for $t = 1, \ldots, T$ **do**
  $\varepsilon_{\text{eff}} \leftarrow \varepsilon_{\text{base}} \cdot \text{clip}(\alpha(\log \|\nabla E(q_{t-1})\|; \theta),\ 0.10,\ 2.50)$
  $p \sim \mathcal{N}(0, I)$; $(q', p') \leftarrow \textsc{Leapfrog}(q_{t-1}, p, \varepsilon_{\text{eff}}, L)$
  Accept $q_t \leftarrow q'$ w.p. $\min(1, e^{-\Delta H})$, else $q_t \leftarrow q_{t-1}$
  if $t \le T_w$ **then**
    $\theta \leftarrow \theta - \eta \nabla_\theta (|\Delta H| - \Delta H^\star)^2$ *(Adam)*
  end if
end for
return $\{q_t\}_{t=T_w+1}^{T}$

approximate fix symmetrises ε as $\frac{1}{2}(\varepsilon_{\text{eff}}(q_0) + \varepsilon_{\text{eff}}(q_L))$. The present experiments use the uncorrected form as a controlled baseline, noting that the bias is small when α varies slowly along a trajectory. Ergodicity holds because $\alpha > 0$ strictly and Gaussian momentum refreshment ensures irreducibility.

Online warm-up training. The MLP is trained for T_w steps by minimising the energy-error loss $\mathcal{L}(\theta) = (|\Delta H(\varepsilon_{\text{eff}}(\theta))| - \Delta H^\star)^2$, where $\Delta H^\star = -\log(0.75) \approx 0.288$ targets 75% acceptance. Because leapfrog is implemented as `jax.lax.scan`, reverse-mode AD propagates through all L steps, requiring Hessian-vector products ($\approx 2\times$ gradient cost during warm-up only). Parameters are updated with Adam (gradient clipping to unit norm); after step T_w the MLP is frozen. Algorithm 1 summarises the full procedure.

3 Experiments

Setup. All samplers run for 50,000 steps from $q_0 = (0, 0)$. RWMH uses $\sigma = 1$; HMC and ALCS-HMC share $\varepsilon_{\text{base}} = 0.2$, $L = 10$ (Rosenbrock) and $\varepsilon_{\text{base}} = 0.1$, $L = 10$ (Neal's funnel); ALCS-HMC uses $T_w = 10{,}000$ warm-up steps with $\eta = 3 \times 10^{-4}$.

Results. Tables 1–2 summarise the key diagnostics. RWMH achieves 50% acceptance on Rosenbrock but its mean $\hat{x} = 0.05$, $\hat{y} = 0.27$ shows the chain never left the origin neighbourhood, confirming local jittering without global mixing. HMC recovers Rosenbrock moments well ($\hat{x} = 1.94$, $\hat{y} = 9.33$) and shows a high 93% acceptance on the funnel, but this reflects an overly conservative step size: the $\text{sd}[x]$ estimate of 7.03 vs. truth 9.49 confirms under-exploration of the wide mouth. ALCS-HMC reaches the 75% acceptance target on both benchmarks by design, and improves over RWMH on Rosenbrock, but its moment estimates indicate it has not yet fully traversed the banana tail or the funnel mouth—consistent with the known bias from the asymmetric step-size proposal discussed in Section **??**.

Table 1: Acceptance rates.

Sampler	Rosenbrock	Neal's funnel
RWMH	50.0%	37.8%
HMC	47.2%	93.2%
ALCS-HMC	**76.4%**	**76.4%**

4 Discussion

What works. The ablation study reveals two robust findings. First, **symmetric step size (Fix B)** is the single most impactful change: on Rosenbrock it raises post-warmup acceptance from 39.3% (asymmetric) to 78.8%, confirming that the detailed-balance bias identified in Section **??** is not merely theoretical—it causes a measurable collapse in practice. On Neal's funnel, Fix B similarly improves stability (loss converges to 0.076 vs. divergence at 35,026 for the asymmetric variant). Second, **very small learning rates** ($\eta = 3 \times 10^{-5}$) produce the most reliable training: the loss drops from 1.98

Table 2: Sample moments vs. analytic ground truth. Bold = closest to truth per column.

		RWMH		HMC		ALCS-HMC	
Dist.	Var (truth)	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
Rosen-2*brock	x (1.0 / 3.16)	0.05	0.64	**1.94**	2.27	0.49	**2.16**
	y (11.0 / 6.50)	0.27	0.66	**9.33**	**7.00**	4.90	4.72
Neal's 2*funnel	v (0.0 / 3.00)	—	—	**0.45**	**2.90**	—	—
	x (0.0 / 9.49)	—	—	$-$**0.45**	7.03	-0.26	3.30

to 1.85 and acceptance reaches 84.2%, whereas larger rates ($\eta \geq 10^{-4}$) cause loss explosion to 10^{22}–10^{23}, suggesting the energy-error loss landscape is extremely steep near the solution.

Where it struggles. The trained MLP provides no consistent advantage over a frozen or fixed-α controller: on Rosenbrock, fixed $\alpha = 1$ achieves 83.1% acceptance while the trained MLP produces diverging losses (to ∞), indicating the online training objective is not reliably solved under the default hyperparameters. The ESJD loss (Q5) saturates the α clip at 2.50 on the funnel, collapsing acceptance to 89% with $\hat{\sigma}[x]$ far from truth, suggesting the move-distance reward overwhelms the acceptance penalty. The small base step $\varepsilon = 0.05$ gives 99% acceptance but means the chain moves slowly; larger values ($\varepsilon \geq 0.2$) cause loss spikes, so the usable range is narrow.

Future directions. The most pressing fix is replacing the online energy-error loss with a stable offline or meta-learning scheme so that the MLP parameters actually converge. Combining Fix B (symmetric step) with a lower learning rate (3×10^{-5}) and a smaller base step ($\varepsilon = 0.05$) appears to be the most promising configuration for a next iteration.

5 AI Collaboration

Reflect on your use of AI assistants:

Tools and interfaces. Two AI assistants were used via chat interfaces. **Google Gemini** was consulted at the project-planning stage to identify the specific vulnerabilities of RWMH and HMC on the two benchmarks and to design the high-level ALCS-HMC concept. Gemini also drafted the prompts used to direct **Claude (Anthropic)** for all implementation work, including adversarial critique prompts intended to stress-test the design. Claude was then used exclusively for code generation, debugging, and paper writing throughout the rest of the project.

Effective prompting strategies. The most effective pattern was *context-first, question-second*: providing Claude with the full current state of the code, the specific error or failure mode, and a clearly scoped question (e.g., "the acceptance rate is nan; here are the four suspected root causes—fix each one"). Gemini's adversarial prompts—which challenged the reversibility of the position-dependent step size, the degeneracy of the energy-error loss, and the second-order derivative cost—were especially productive because they forced precise theoretical justifications and led directly to Fix A/B and the ESJD loss formulation.

Where AI helped vs. where guidance was needed. Claude was highly effective at translating mathematical specifications into correct JAX code (e.g., the `lax.scan` leapfrog, Adam on pytree parameters, the warm-up/freeze pattern with `lax.cond`) and at diagnosing numerical instabilities systematically. Human intervention was needed when the framing of the problem was wrong rather than the code: the original $\alpha \in [0.05, 20]$ clip that caused immediate divergence, the use of `while_loop` inside a differentiated loss, and the interpretation of ablation results all required the user to identify the conceptual issue before Claude could resolve it.

Lessons learned. Dividing AI roles by *abstraction level* worked well: Gemini for strategy and adversarial stress-testing, Claude for implementation and incremental debugging. The key lesson is that AI assistants accelerate iteration but do not replace the need to understand the underlying

theory—the most consequential bugs (broken detailed balance, loss degeneracy) were identified through theoretical reasoning, not by running more code. Treating AI-generated code as a first draft that requires the same critical review as one's own code, rather than as a finished product, was the most important working habit developed over this project.