JP Morgan Report
Hamiltonian Neural Network and Hamiltonian Monte Carlo
Name: Xingguang (Ray) Jin
Email: RayJIN@link.cuhk.edu.hk

1 Literature Review

Due to the high computational cost of Bayesian inference, machine learning has garnered considerable interest in accelerating this process. The following Table 1 summarizes existing methods for solving Bayesian inference problems.

HMC and HNN share common features, as both utilize Hamiltonian dynamics to propose new states, making them particularly efficient in high-dimensional spaces. However, the HMC class of algorithms can require numerous numerical gradients of the target density, which can be computationally expensive. In contrast, HNN combines HMC with neural networks, allowing it to model complex relationships and uncertainties. This approach is efficient in high dimensions and can learn complex posterior distributions without the need for numerical gradient computation.

Table 1: Overview of different algorithms

Methods	Features
Markov chain Monte Carlo (MCMC)	Simple to implement with any given distribution
Random-walk MCMC	Weak perform in high-dimensional spaces
Hamiltonian Monte Carlo (HMC)	Require numerous numerical gradients
Hamiltonian Neural Network (HNN)	No numerical gradients
Latent Hamiltonian neural networks (L-HNNs)	Reduced integration errors

2 Hamiltonian dynamics

If $\mathbf{p}$ and $\mathbf{q}$ represent the position and momentum vectors, respectively, the Hamiltonian system is defined as:

$$H(\mathbf{q}, \mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p}),$$

and Hamiltonian equation satisfies the following

$$\frac{d\mathbf{q}}{dt} = \frac{\partial H}{\partial \mathbf{p}}, \quad \frac{d\mathbf{p}}{dt} = -\frac{\partial H}{\partial \mathbf{q}},$$

where $(\mathbf{q}, \mathbf{p})_i = (q_i, p_i), i = 1 \dots M$.

More interestingly,

$$\frac{\partial H}{\partial t} = \frac{\partial H}{\partial \mathbf{p}} \frac{d\mathbf{p}}{dt} + \frac{\partial H}{\partial \mathbf{q}} \frac{d\mathbf{q}}{dt} = 0,$$

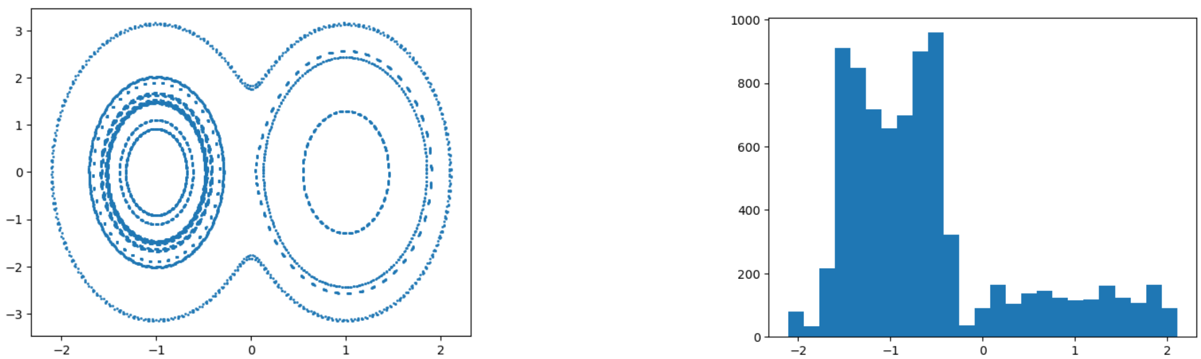
which means the Hamiltonian system keeps the output constant if it moves in the direction $(\frac{\partial H}{\partial \mathbf{p}}, -\frac{\partial H}{\partial \mathbf{q}})$. That also explains why there is a minus sign in the Hamiltonian equation.

3 Hamiltonian Monte Carlo (HMC)

Open *hmc.ipynb* to visualize HMC to sample from 1D Gaussian mixture density. The following 1D Gaussian mixture density was considered :

$$f(q) \propto 0.5\exp(-\frac{(q-1)^2}{2\times 0.35^2}) + 0.5\exp(-\frac{(q+1)^2}{2\times 0.35^2}) \tag{1}$$

To generate the training data, I chose the Hamiltonian dynamic for 20 samples, with $T = 20$ units. Figure 1 reproduces the same figure in [2].



((a)) ((b))

Figure 1: (a) Phase space plot corresponding to different energy levels. (b) Histogram of samples from a $1D$ Gaussian mixture density using traditional HMC.

4 Hamiltonian Neural Network (HNN)

Continuously open *hnntest.ipynb* to visualize HNN to sample from 1D Gaussian mixture density. The definition of the loss function of HNN should be updated to the following formula:

$$L = \frac{1}{M}(\sum_{i=1}^{M}\int(\frac{\partial H_\theta}{\partial p}(t) - \frac{\Delta q}{dt}(t))^2 dt + \int(\frac{-\partial H_\theta}{\partial p}(t) - \frac{\Delta q}{dt}(t))^2 dt),$$

compared with the traditional neural network. Figure 2 is the histogram of samples from a 1D Gaussian mixture density using HNN, which is the same result in [2].

I also chose 2D Neal's funnel density for this analysis and successfully reproduced the results shown in Figure 8 of [2]. For any other figures, you only need to modify the density functions accordingly.

The following $2D$ Neal's funnel density was considered :

$$f(q_1, q_2) \propto \begin{cases} q_1 = \mathcal{N}(0,3) \\ q_2 = \mathcal{N}(0, \exp^{q_1}). \end{cases} \tag{2}$$

- Open $2d\_neal\_funnel\_density.py$.

  Set the step to be 25000. It takes 12 hours to obtain the Figure 3 and Figure 4.

- Include the Hamiltonian of the required probability distribution in the $functions.py$ and set the step to be 1000. It takes 20 mins to obatin the Figure 5 and Figure 6.

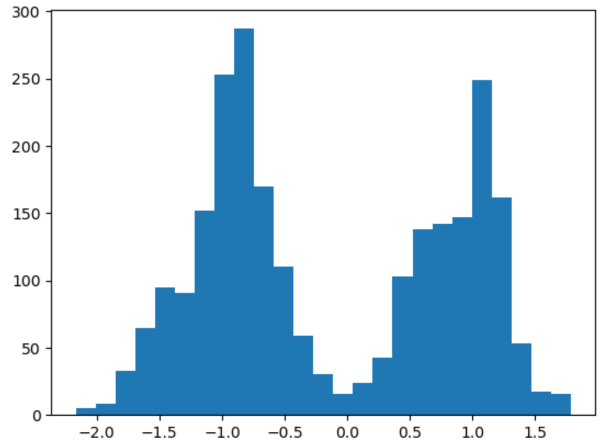


Figure 2: Histogram of samples from a 1D Gaussian mixture density using HNN

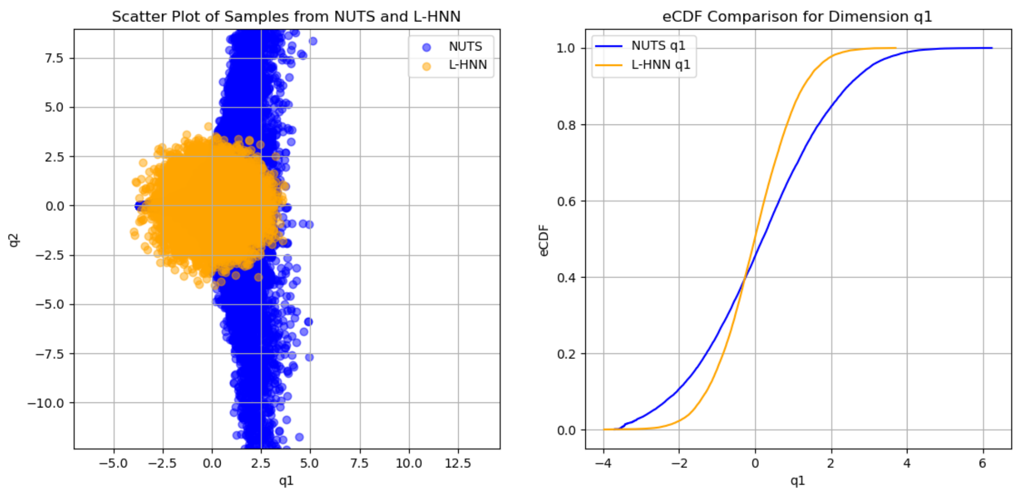


Figure 3: (Left) Scatter plot comparison of samples generated from a 2-D Neal's funnel density, using NUTS and L-HNNs; (Right) Comparison of the eCDF plots for the dimensions q_1.

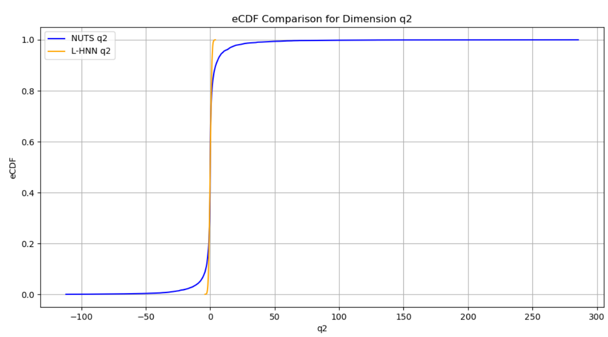


Figure 4: Comparison of the eCDF plots for the dimensions q_2.

5 Extended Hamiltonian Neural Network (E-HNN)

The definition of the extended Hamiltonian in [1]:

$$H(\theta, \rho, u, p) = -\log \rho(\theta) - \log \hat{p}(y|\theta, u) + \frac{1}{2}\{\rho^T \rho + u^T u + p^T p\}.$$

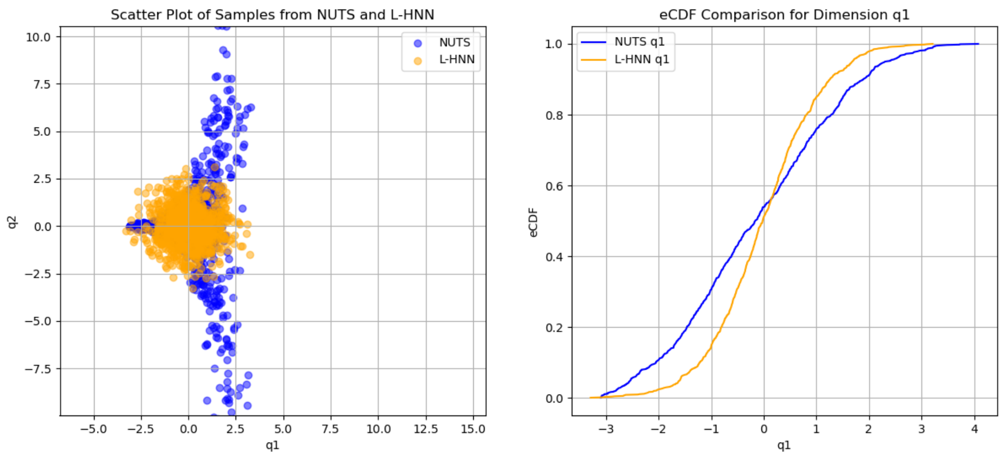


Figure 5: (Left) Scatter plot comparison of samples generated from a 2-D Neal's funnel density, using NUTS and L-HNNs; (Right) Comparison of the eCDF plots for the dimensions q_1.

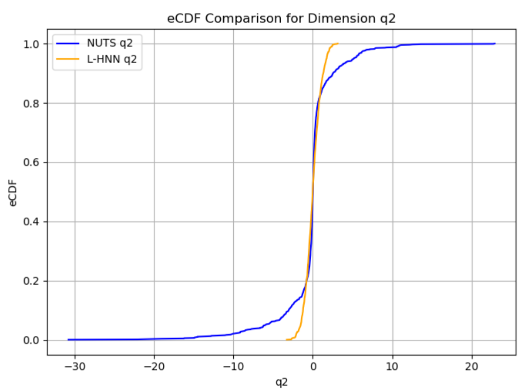


Figure 6: Comparison of the eCDF plots for the dimensions q_2.

5.1 Initial Data

$$\theta_0^i, u_0^i = \theta^{i-1}, u^{i-1}; \rho(0), p(0) \sim \mathcal{N}(0, I_d)$$

Sample: for $j = 1, \ldots, N$, the number of trajectory

$$(\theta^j(t+h), \rho^j(t+h), u^j(t+h), p^j(t+h))^T = \Phi^A_{h/2} \circ \Phi^B_h \circ \Phi^A_{h/2}(\theta^j(t), \rho^j(t), u^j(t), p^j(t)^T)^T.$$

$$\theta^j(t_i) = \frac{\theta^j(t_{i+1}) - \theta^j(t_i)}{\Delta t}.$$

5.2 Loss function

The loss function of E-HNN should be defined similarly:

$$L = \frac{1}{M}(\sum_{i=1}^{M} \int (\frac{d\theta}{\partial t}(t) - \nabla_\rho H(t))^2 dt + \int (\frac{d\rho}{dt}(t) + \nabla_\theta H)^2 dt + \int (\frac{du}{dt}(t) - \nabla_p H)^2 dt + \int (\frac{dp}{dt}(t) + \nabla_u H)^2 dt),$$

The original Algorithm 1 (Pseudo-marginal HMC) in [1] is updated by using the HNN.

5.3 Algorithm

Algorithm 1 Pseudo-marginal Hamiltonian Neural Network

Samples: M; Starting sample:$\{\theta^0, \rho^0, u^0, p^0\}$; End time for trajectory:T; Steps:N; Dimensions: d
for $i = 1 : M$ **do**
    $\theta(0) = \theta^{i-1}$
    $\rho(0) \sim \mathcal{N}(0, I_d)$
    $p(0) \sim \mathcal{N}(0, I_d)$
    $u(0) = u^{i-1}$
    Compute $\{\theta^*, \rho^*, u^*, p^*\} = \{\theta(T), \rho(T), u(T), p(T)\}$ with Hamiltonian Neural Network via leapfrog or operator splitting equation (19) in [1]
    $\alpha = \min\{1, \exp(H(\theta^i, \rho^i, u^i, p^i) - H(\theta^*, \rho^*, u^*, p^*))\}$
    With probability α, set $\{\theta^i, \rho^i, u^i, p^i\} \leftarrow \{\theta^*, \rho^*, u^*, p^*\}$
end for

6 Applications

6.1 Risk Management

Risk is essentially measuring change of captial upon change of some other parameters. Given a portfolio, a confidence level, and a simulation method, we will be able to calculate the probability that less than $(1-C)\%$, the portfolio is going to loss more than the VaR.

$$\mathbb{P}(L > \text{VaR}) \leq 1 - C,$$

where L is the loss.

6.2 Simulate stock price-volatility pairs

Quantitative descriptions of finance rely heavily on PDEs and SDEs. These are natural choices as stock prices can depend on several variables and change over time randomly. Consider the celebrated Heston model (S_t, V_t), where S_t is the asset price at time t and V_t is the instantaneous variance (volatility squared) at time t.

$$\begin{aligned} dS_t &= \mu S_t dt + \sqrt{V_t} S_t dW_{S,t} \\ dV_t &= \kappa(\theta - V_t)dt + \xi \sqrt{V_t} dW_{V,t} \end{aligned}$$

we can simulate Heston model (S_t, V_t) in Figure 7 with a Hamiltonian dynamics $H(\mathbf{q}, \mathbf{p})$ such

$$H(\mathbf{q}, \mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p}),$$

and Hamiltonian equation satisfies the following

$$\frac{d\mathbf{q}}{dt} = \frac{\partial H}{\partial \mathbf{p}}, \quad \frac{d\mathbf{p}}{dt} = -\frac{\partial H}{\partial \mathbf{q}},$$

where $(\mathbf{q}, \mathbf{p})_i = (q_i, p_i), i = 1 \ldots M$.

Figure 7: The top panel shows the simulated stock price over time, while the bottom panel shows the instantaneous volatility.

References

[1] Johan Alenlöv, Arnaud Doucet, and Fredrik Lindsten. Pseudo-marginal hamiltonian monte carlo. *Journal of Machine Learning Research*, 22, 2021.

[2] Somayajulu L.N. Dhulipala, Yifeng Che, and Michael D. Shields. Efficient bayesian inference with latent hamiltonian neural networks in no-u-turn sampling. *Journal of Computational Physics*, 492:112425, 2023.