Hamiltonian-Inspired Optimization in Neural Networks

Javier Marín
javier@jmarin.info

The optimization problem

Optimization in machine learning seeks to find parameters θ^* that minimize an objective function

$$\theta^* = \text{argmin}\, f(x) \quad \text{for } \theta \in \Omega$$

Key challenges include non-convexity, high dimensionality, and ill-conditioning of the objective function.

Common approaches:

- First-order methods (e.g., Gradient Descent)
- Second-order methods (e.g., Newton's method)
- Momentum-based methods (e.g., Nesterov Accelerated Gradient)
- Adaptive methods (e.g., Adam)
- Hamiltonian-inspired methods (leveraging principles from classical mechanics)

Each approach offers trade-offs between computational efficiency, convergence speed, and ability to navigate complex optimization landscapes.

Non-convex optimization

The generic form of an analytic optimization problem is the following:

$$\min_{x \in \mathbb{R}^p} f(x) \quad \text{for } x \in \mathcal{C}$$

Where x is the variable of the problem, $f: \mathbb{R}^p \longrightarrow \mathbb{R}$ is the objective function of the problema and $\mathcal{C} \in \mathbb{R}^p$ is the constraint. While the constraint set $\mathcal{C}$ is often assumed to be convex, we extend our consideration to non-convex sets. This extension is important for many machine learning problems.

In traditional optimization, we use the gradient operator $\nabla f(x)$ and the projection operator $\prod \mathcal{C}(\cdot)$. These are fundamental to our approach, but we will modify their use in the Hamiltonian framework.

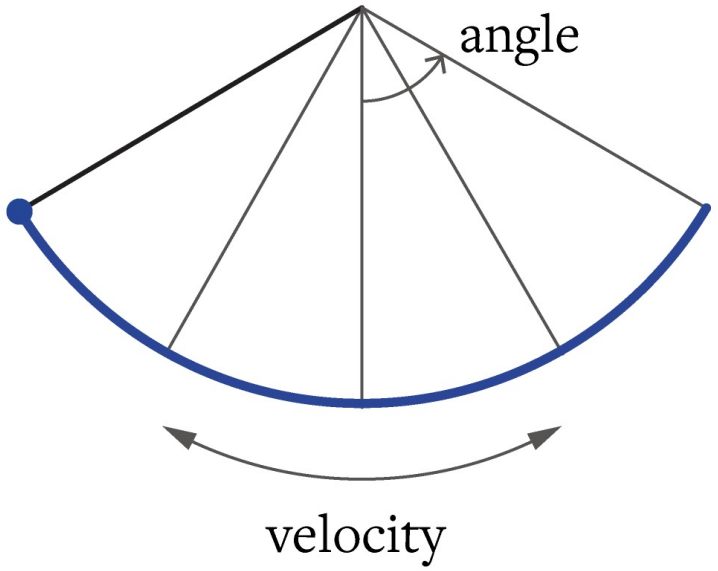
The standard Projected Gradient Descent (PGD) algorithm for solving the generic form above is:

$$x(t+1) = \prod \mathcal{C}\,(x(t) - \eta \nabla f(x(t))$$

wehere η the step size. Our Hamiltonian approach will build upon this foundation.

Introduction to Hamiltonian Mechanics

Hamiltonians in physics describe systems that conserve total energy, potentially leading to more stable optimization trajectories. In quantum mechanics, Hamiltonians govern the evolution of wavefunctions, exploring all possible states. Similarly, in optimization, this could lead to better exploration of the parameter space.



In a frictionless pendulum the total energy, it is the sum of potential and kinetic energy is conserved trough its evolution.

Introduction to Hamiltonian Mechanics

A **Hamiltonian** can be defined with the following $2n$ ordinary differential equations of motion:

$$\dot{q} = H_p \ , \qquad \dot{p} = -H_q$$

$$\dot{q}_l = \frac{\partial H}{\partial p_l}(t, q, p) \ , \qquad p_i = -\frac{\partial H}{\partial q_i}(t, p, q)$$

where $H = H(t, q, p)$ is the Hamiltonian, q and p are the position and momentum vectors of a mechanical system with n degrees of freedom, and t is the time.
In isolated systems, the Hamiltonian system $H(q, p, t)$ remains constant over time, representing the "total energy" of the system

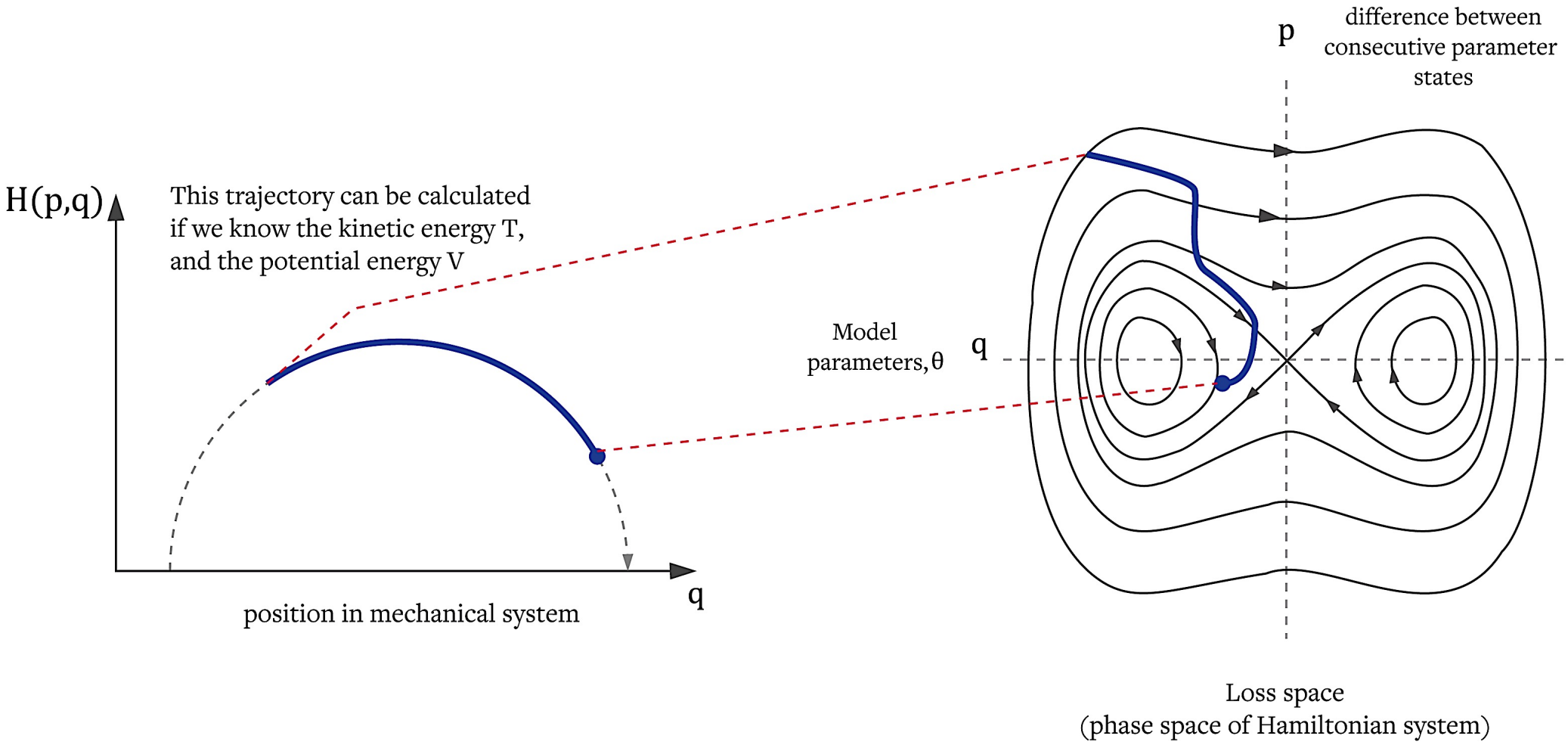
$$\frac{dH}{dt} = \frac{\partial H}{\partial t} + \{H, H\} = \frac{\partial H}{\partial t} = 0$$

where $\{\ ,\}$ denotes the Poisson bracket operator.

Hamiltonian Systems in ML

Hamiltonian system in mechanics

Neural Network Optimization



Symplectic Geometry: The Hidden Structure of Dynamical Systems

Neural network optimization can be analyzed as a trajectory in parameter space of a Hamiltonian system. Equations of motion in a Hamiltonian system:

$$\frac{dq}{dt} = \frac{\partial H}{\partial p} \quad / \quad \frac{dp}{dt} = \frac{\partial H}{\partial q}$$

In most practical applications, we can't solve these equations analytically. Symplectic integrators are numerical methods specifically designed to solve Hamiltonian systems while preserving their key geometric properties, particularly the symplectic structure. This is why symplectic geometry is important for us.

We can define a Hamiltonian in an optimization space H: $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ as $H(q,p) = T(p) - V(q)$, where q represents the current state of the model parameters (θ), analogous to position in mechanical systems, and p is the difference between consecutive parameter states, $p_i = q_{\{i+1\}} - q_i$. $T(p)$ is the "kinetic energy" term representing the cost of changing the model parameters, and $V(q)$ is the "potential energy" term representing the loss function of the current model state.

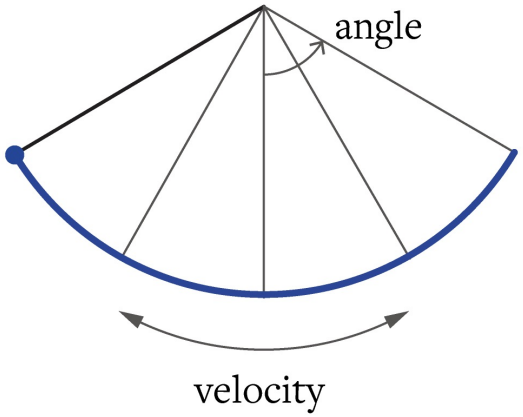
Symplectic Geometry

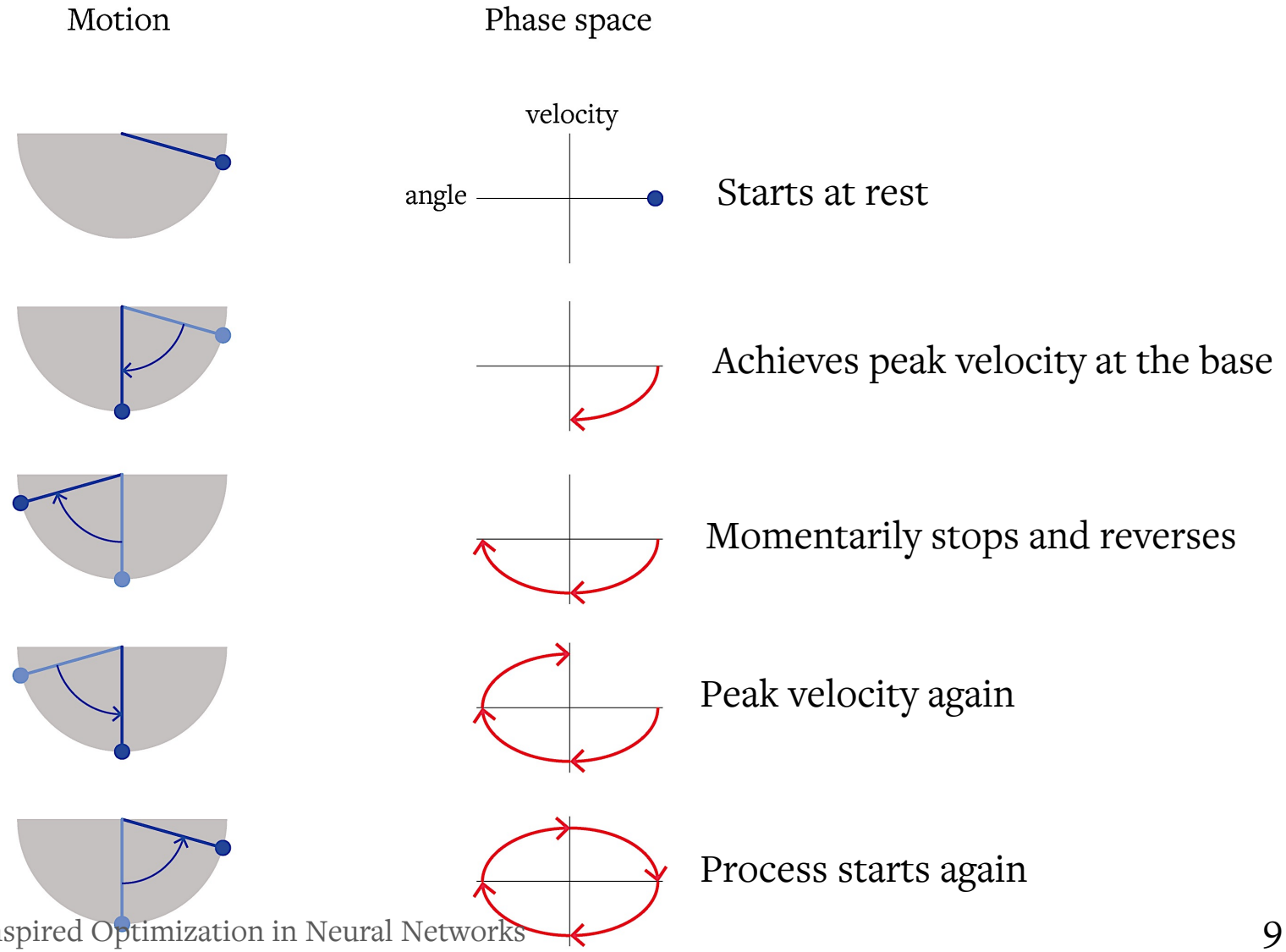
Symplectic structures are fundamental geometric objects in differential geometry and classical mechanics, and support Hamilton's equations of motion by explaining the connection between position and momentum in physical systems. In simple terms, **symplectic structures are specific rules that define how things move in physics, similar to an equation for motion**. **Poincaré's Theorem** states that any solution to a Hamiltonian system is a symplectic flow, and it can also be shown that any symplectic flow corresponds locally to an appropriate Hamiltonian system.

Symplectic integrators are numerical methods specifically formulated to solve Hamiltonian systems while preserving their fundamental geometric features, particularly the symplectic structure. In simpler terms, **these tools help scientists make very accurate predictions about how things move in space or in other systems where energy is conserved, even when they're looking far into the future.** They're like the difference between a map that stays accurate for a short walk and one that can guide you accurately on a journey around the world.

Symplectic Geometry

A frictionless pendulum is one of the most basic forms of a symplectic space. Velocity and angle are the two components that describe the movements of a pendulum. We can map this in a 2D space as a trajectory.





Symplectic Euler method

In local coordinates (q_i, p_i), a **standard symplectic form** can be written as:

$$\omega = \sum_i dq_i \wedge dp_i$$

The simpler form of a symplectic integrator that aligns more closely with our implementation would be the **Symplectic Euler method**. This is a first-order symplectic integrator and is one of the simplest forms of integration. For a Hamiltonian system with position q and momentum p, the Symplectic Euler method can be written as:

$$p_{t+1} = p_t - \Delta t \times \frac{\partial H}{\partial q}(q_t, p_{t+1})$$

$$q_{t+1} = q_t - \Delta t \times \frac{\partial H}{\partial p}(q_t, p_{t+1})$$

Where Δt is the time step (analogous to the learning rate in optimization).

Hamiltonian Optimizer

For optimization functions that are α-strongly **convex** and β-strongly **smooth**, we have:

$$\frac{\alpha}{2}\|x-y\|^2 \leq f(y) - f(x) - \langle \nabla f(x), y-x \rangle \leq \left(\frac{\beta}{2}\right)\|x-y\|^2$$

Although our approach focuses on **non-convex functions**, these characteristics enhance our analysis of function landscapes.

Traditional convergence analysis for PGD shows that for convex, Lipschitz functions:

$$\left(\frac{1}{T}\right)\sum^{t} f(x(t)) - f(x^*) \leq O\left(\frac{1}{\sqrt{T}}\right)$$

We will adapt **similar** approaches to analyze our Hamiltonian method.

Hamiltonian Optimizer

For non-convex problems, PGD can be modified to use local properties such as restricted strong convexity. Our Hamiltonian method will provide an alternative approach to handling non-convexity. We have already seen the expression for functions that are α-strongly convex and β-strongly smooth. This concept will underpin our understanding of the local features of our potential energy function.

The **optimization** problem is **reformulated** as finding the stationary points of the Hamiltonian

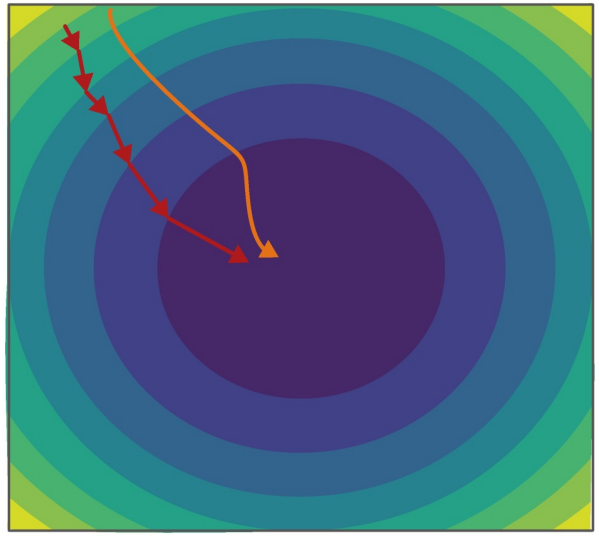
$$\frac{\partial x}{\partial p} = -\frac{dp}{dt} \qquad \frac{\partial H}{\partial p} = -\frac{dx}{dt}$$

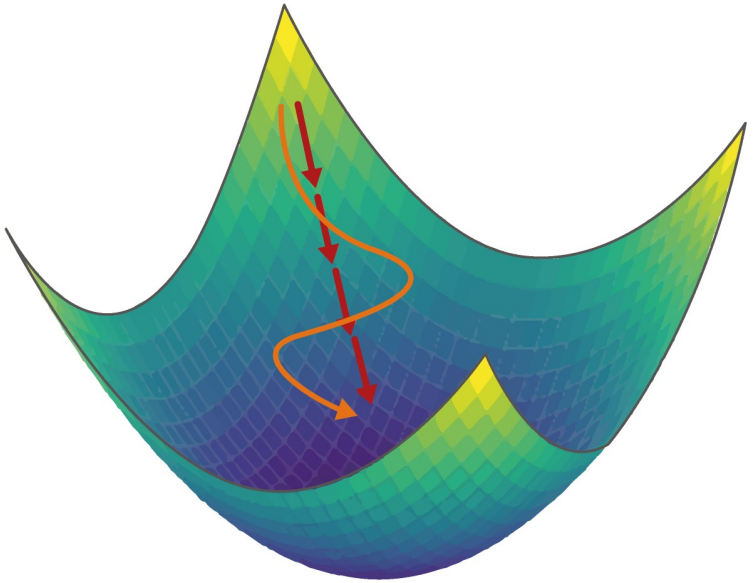
Note: While not directly used, the concept of optimizing over multiple variables is relevant to our approach, as we will alternate between position and momentum updates.

From Gradient Optimizer to Hamiltonian Optimizer

Function value $f(x) \rightarrow$ Potential energy, $V(x)$

Rate of parameter change $d\theta \rightarrow$ Kinetic energy, $T(x)$





Momentum based approach translates to the algorithm being able to ″roll past″ small local minima in the loss landscape, potentially finding better global or local minima that simple gradient descent might miss. The conservation of the Hamiltonian (total energy) ensures that the system maintains this exploratory behavior throughout the optimization process, unlike in some other methods where the exploration gradually decreases.

→ Step-by-step Gradiend descent → Hamiltonian (momentum based)

Hamiltonian (Symplectic) Optimizer

Algorithm 1 Hamiltonian (Symplectic) Optimizer

Require: Learning rate η, momentum coefficient β, epsilon ε
for each iteration **do**
  for each parameter θ in model parameters **do**
    $g \leftarrow \nabla L(\theta)$ // Compute gradient
    $v \leftarrow$ state[θ]['momentum'] // momentum

    // Update momentum
    $v \leftarrow \beta \times v + (1 - \beta) \times g$

    // Compute Hamiltonian
    $K \leftarrow 0.5 \times \|v\|^2$ // Kinetic energy
    $V \leftarrow 0.5 \times \|g\|^2$ // Potential energy
    $H \leftarrow K + V$ // Hamiltonian

    // Update parameter
    $\theta \leftarrow \theta - \eta \times v / (\sqrt{H} + \varepsilon)$

    // Store updated momentum
    state[θ]['momentum'] $\leftarrow v$
  end for
end for

Momentum update:

$$m_t = \beta \times m_{(t-1)} + (1 - \beta) \times g_t$$

- We add a momentum decay factor β

Hamiltonian:

- We separate the computation of the Hamiltonian as kinetic + potential energy

Parameter update:

$$\theta_t = \theta_{t-1} - \frac{\eta \times m_t}{\sqrt{H_t + \varepsilon}}$$

- We normalize by the square root of the Hamiltonian $\sqrt{H_t + \varepsilon}$

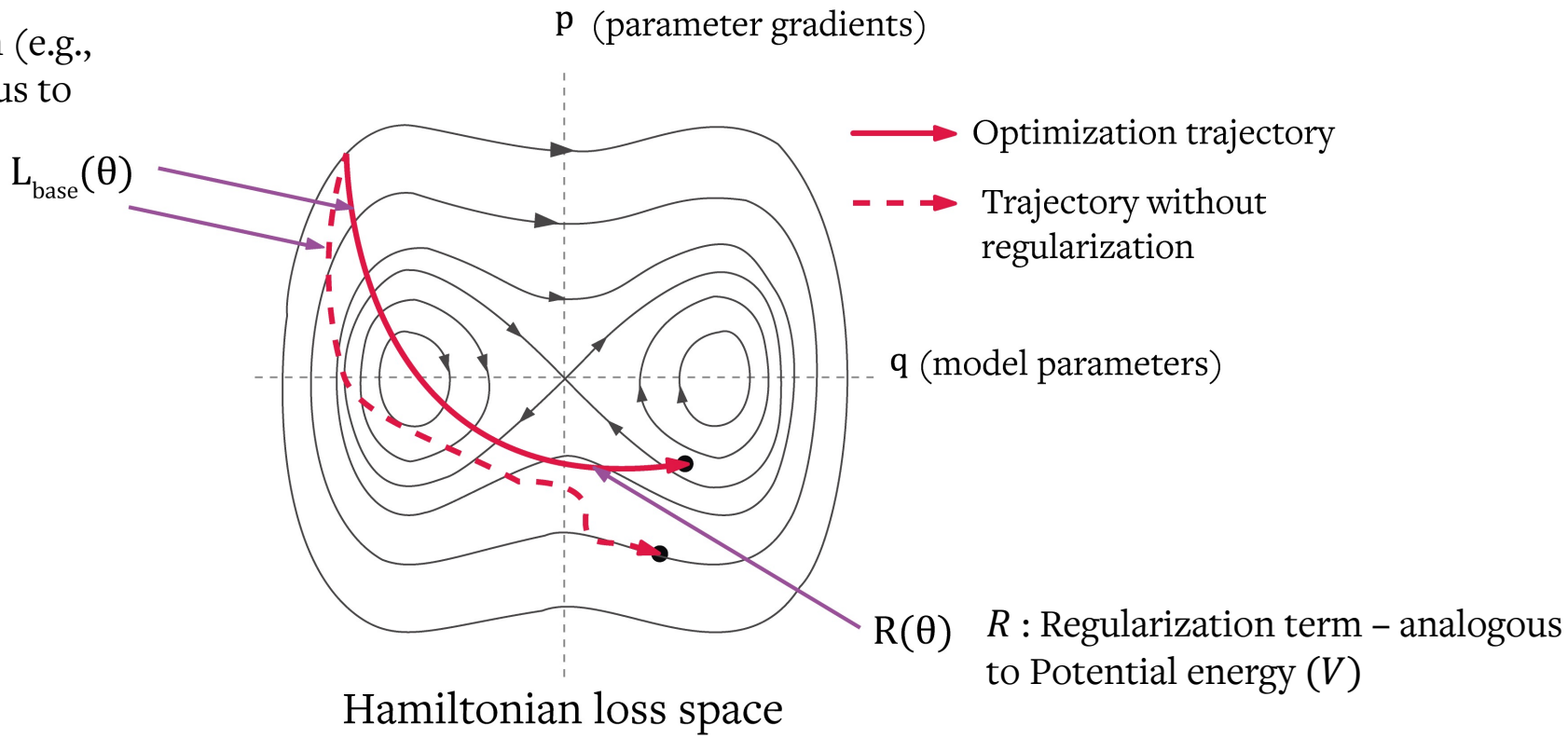
m – momentum, the same as p in $H(q,p)$
H – Hamiltonian, $H(q,p)$
θ – parameters, the same as q in $H(q,p)$
η – learning rate
ε – small constant for stability
g – gradient

Hamiltonian Loss Function

Hamiltonian loss is the combination of a standard loss $L_{base}\ (\theta)$, plus a regularization term, $\lambda \times R(\theta)$.

$$H_{loss}(\theta) = L_{base}\ (\theta) + \lambda \times R(\theta)$$

L_{base} : Base loss function (e.g., cross-entropy) - analogous to kinetic energy (T)



Hamiltonian Loss Function

Potential energy in Hamiltonian Systems

Regularization term

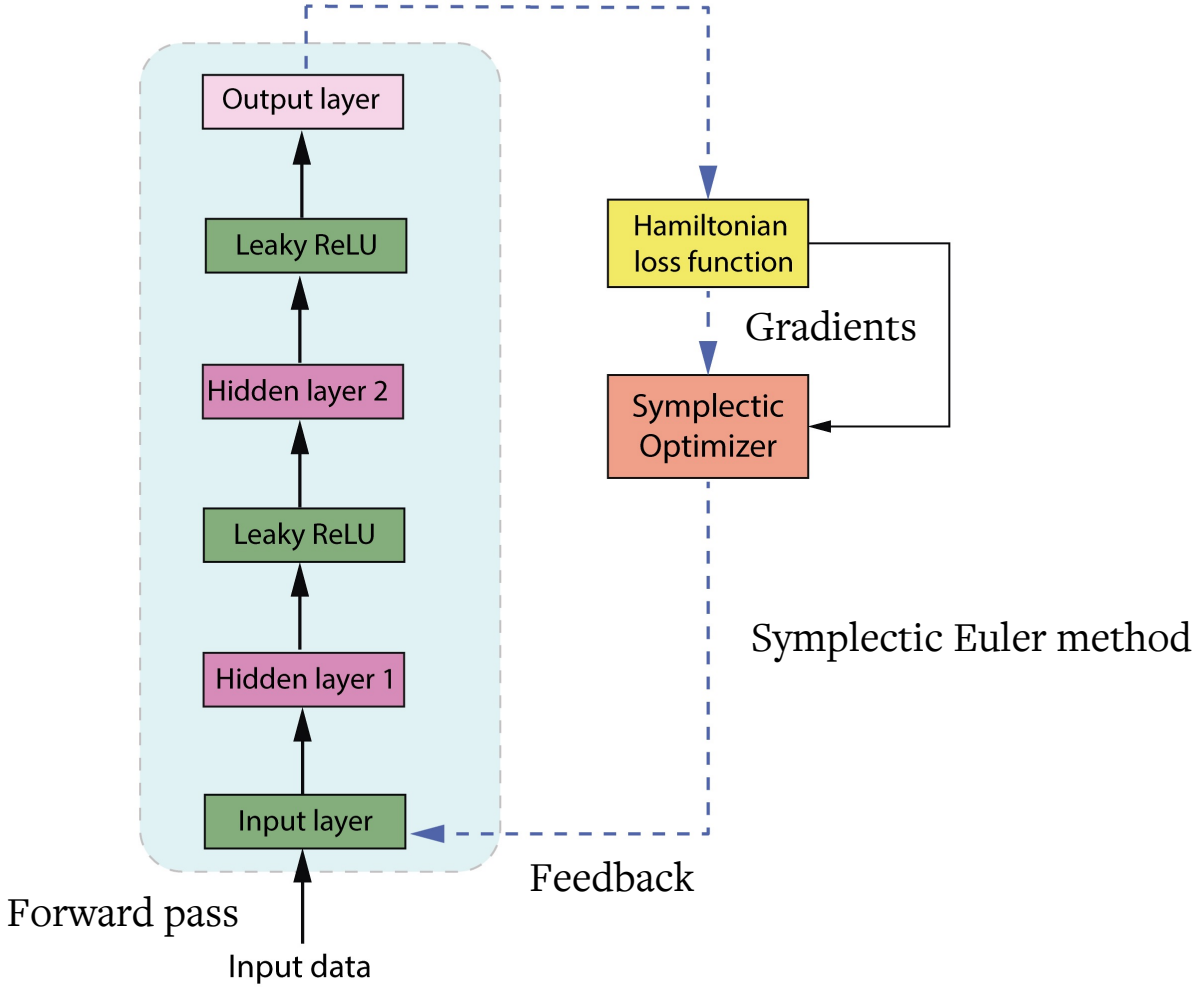
$$T(p) = \frac{1}{2}\|p\|^2 \longrightarrow R(\theta) = \frac{1}{2}\|\theta_i\|^2$$

*$R(\theta)$ penalizes large parameter values, promoting smoother solutions. In mechanical systems, high potential energy can lead to large oscillations. By penalizing high potential energy, we're essentially damping these oscillations, leading to smoother trajectories in parameter space.

Algorithm 2 Hamiltonian Loss Function

Require: Model outputs, true labels, model parameters θ, regularization coefficient λ
  $L_{base} \leftarrow$ CrossEntropy(outputs, labels) // Compute base loss
  $R \leftarrow 0.5 \times \sum\|\theta_i\|^2$ // Compute regularization term
  for each parameter θ_i in model parameters **do**
    $R \leftarrow R + 0.5 \times \|\theta_i\|^2$
  end for
  $H_{loss} \leftarrow L_{base} + \lambda \times R$ // Compute total Hamiltonian loss
return H_{loss}

Combining Hamiltonian Optimizer and Hamiltonian Loss



Experimental results I

Standard GPT-2 model vs. Hamiltonian-inspired model

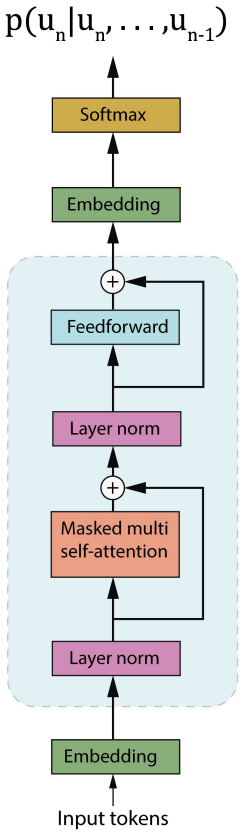
Key Findings:

1. Hamiltonian model showed improved performance in multi-hop reasoning tasks
2. Lower and more stable Hamiltonian energy profiles for valid reasoning chains
3. Smoother trajectories with lower curvature in embedding space for valid chains
4. Better conservation of angular momentum-like quantities in valid reasoning

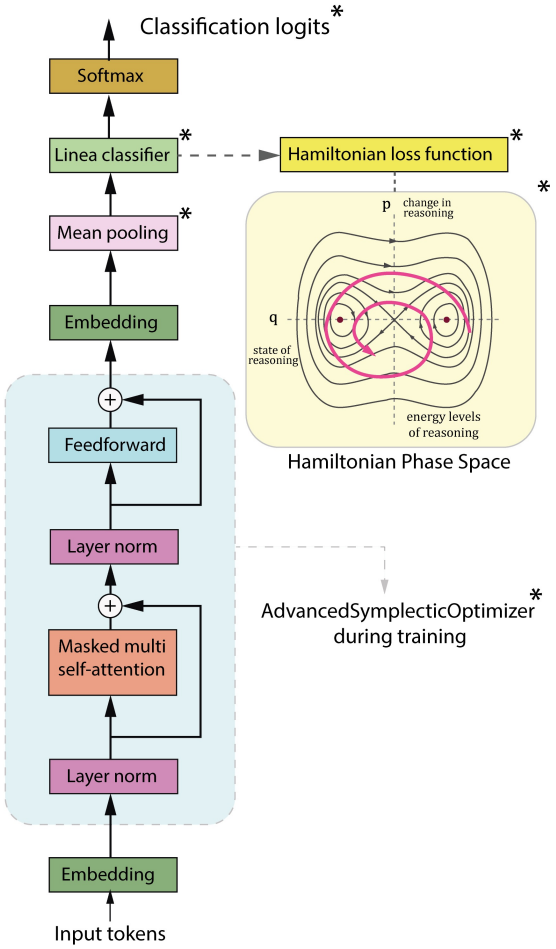
Advantages of Hamiltonian Approach:

1. Enhanced ability to distinguish valid from invalid reasoning paths
2. More efficient balance between cognitive progression and relevance
3. Potential for guiding AI systems towards more effective reasoning strategies

Standard GPT-2 model



Hamiltonian model

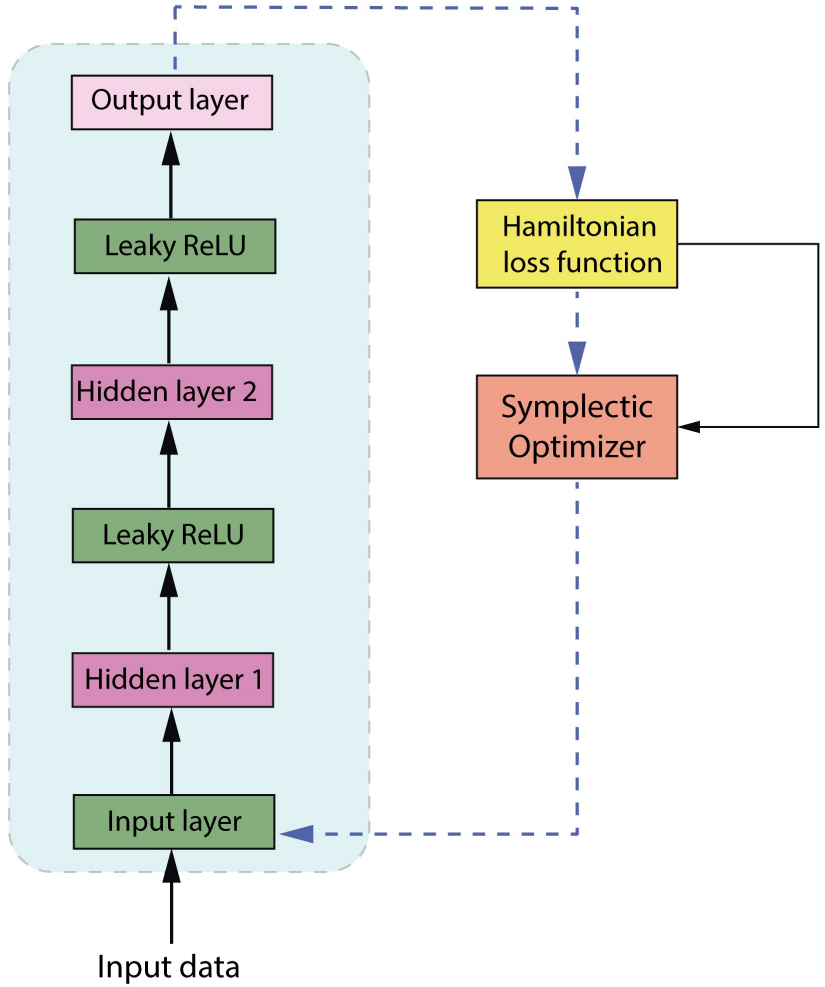


Experimental results II

The experiment compares this approach to XGBoost using the Freddie Mac Single-Family Loan-Level Dataset (SFLLD) across different time horizons (12, 36, and 60 months).

	FM12 (12 months)		FM36 (36 months)		FM60 (60 months)	
Hamiltonian inspired approach						
Mean accuracy	0.8047	∓0.0021	0.7638	∓0.0010	0.6975	∓0.0003
Mean F1 Score	0.8012	∓0.0000	0.7608	∓0.0008	0.6968	∓0.0003
Mean AUC score	0.8027	∓0.0000	0.764	∓0.0000	0.6974	∓0.0000
Mean Precision	0.8201	∓0.0000	0.777	∓0.0000	0.6994	∓0.0000
Mean Recall	0.8381	∓0.0001	0.7638	∓0.0000	0.6973	∓0.0000
XGBoost algorithm						
Accuracy	0.9871		0.9663		0.9319	
Precision	0.9948		0.9831		0.9796	
Recall	0.9827		0.9663		0.9319	
F1 Score	0.9882		0.9745		0.9542	
AUC	0.6072		0.6221		0.6665	

1. The Hamiltonian approach achieves superior Area Under the Curve (AUC) scores across all time horizons, indicating better discriminative power and ranking ability.
2. XGBoost performs better in traditional metrics like accuracy, precision, and recall.
3. The Hamiltonian method demonstrates more consistent performance over time, suggesting better generalization to future, unseen data.



Future directions

1. Explore applications in other complex ML tasks, such as reinforcement learning and generative models.

2. Research the method's scalability to larger, more complex neural network architectures.

3. Develop theoretical safe-guards for convergence and optimality in non-convex settings.

4. Integrate with other optimization techniques to create hybrid approaches.

5. Study the interpretability aspects of Hamiltonian-based optimization trajectories.

6. This Hamiltonian-inspired approach opens new opportunities for optimization in machine learning, bridging physics and AI in promising ways.