# Hyperverlet: A Symplectic Hypersolver for Hamiltonian Systems

Mathiesen, Frederik Baymler; Yang, Bin; Hu, Jilin

**DOI**
10.1609/aaai.v36i4.20381

**Publication date**
2022

**Document Version**
Final published version

**Published in**
Proceedings of the AAAI Conference on Artificial Intelligence

# Hyperverlet: A Symplectic Hypersolver for Hamiltonian Systems

**Frederik Baymler Mathiesen**[1,2], **Bin Yang**[2], **Jilin Hu**[2*]

[1]Delft University of Technology, the Netherlands
[2]Aalborg University, Denmark
frederik@baymler.com, {byang, hujilin}@cs.aau.dk

## Abstract

Hamiltonian systems represent an important class of dynamical systems such as pendulums, molecular dynamics, and cosmic systems. The choice of solvers is significant to the accuracy when simulating Hamiltonian systems, where symplectic solvers show great significance. Recent advances in neural network-based hypersolvers, though achieve competitive results, still lack the symplecity necessary for reliable simulations, especially over long time horizons. To alleviate this, we introduce Hyperverlet, a new hypersolver composing the traditional, symplectic velocity Verlet and symplectic neural network-based solvers. More specifically, we propose a parameterization of symplectic neural networks and prove that hyperbolic tangent is r-finite expanding the set of allowable activation functions for symplectic neural networks, improving the accuracy. Extensive experiments on a spring-mass and a pendulum system justify the design choices and suggest that Hyperverlet outperforms both traditional solvers and hypersolvers.

## Introduction

Systems identification of frictionless dynamical systems such as pendulums and molecular dynamics through deep learning has seen a surge in recent years, particularly Hamiltonian Neural Networks (Greydanus, Dzamba, and Yosinski 2019). To simulate the temporal evolution of these systems, most of methods are agnostic to the numerical solver or rely on a simple Euler solver. However, as shown in (Chen et al. 2020b) the choice of solver is important to the accuracy.

Recent efforts at improving single-step solvers like Euler solvers include hypersolvers (Shen, Cheng, and Liang 2020; Poli et al. 2020), which proposes to use a neural network to correct the error induced by the time discretization of the base solver. However, for the class of Hamiltonian systems, this structure fails at capturing the energy conservation and symplecticity, both of which are key for reliably predicting the long-term evolution. Our method, Symplectic Hyperverlet, adopts the perspective of hypersolvers with a base solver and a neural network corrector for improving the accuracy. We differ from hypersolvers (Shen, Cheng, and Liang 2020; Poli et al. 2020) by focusing only on Hamiltonian systems and show that a symplectic base solver and neural network is important. To support this goal, we utilize velocity Verlet, a symplectic solver, as the base solver, and augment it with a SympNet-based corrector (Jin et al. 2020). In Figure 1, we compare the architectural differences and the impact on the energy conservation.

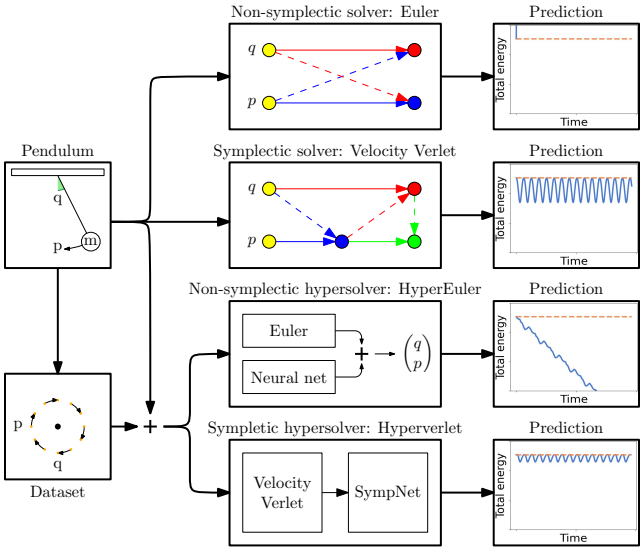


Figure 1: Comparison architectures and impact on energy conservation of Euler, velocity Verlet, HyperEuler, and Symplectic Hyperverlet. $(q, p)$ is position and momentum, i.e. the state of the system. In the prediction plots on the right, the blue, solid line is energy of the prediction computed from $(q, p)$, and the dashed, orange line is the ground truth energy. Symplecticity and deep learning allows hyperverlet to accurately predict the temporal evolution of separable Hamiltonian systems.

To improve the accuracy of our method, we parameterize SympNets based on time invariant properties such as mass, pendulum length, etc. Finally, we improve the accuracy by employing the hyperbolic tangent activation function in the SympNet architecture, and formally prove that it satisfies the conditions under which SympNet is a universal approximation of all symplectic transformations. Table 1 compares the properties of hyperverlet against existing solvers.

*Corresponding Author: Jilin Hu.

| Solver | Hypersolver | Symplectic | Convergent |
|---|---|---|---|
| Euler | | | ✓ |
| HyperEuler | ✓ | | ✓ |
| RK4 | | | ✓ |
| Velocity Verlet | | ✓ | ✓ |
| FR4 | | ✓ | ✓ |
| SympNet | ✓ | ✓ | |
| Hyperverlet | ✓ | ✓ | ✓ |

Table 1: Comparison of solvers by being a hypersolver, symplectic, and convergent.

To summarize, we make the following contributions:

- Propose hyperverlet that interleaves traditional symplectic solvers and parameterized SympNet to improve accuracy while preserving the symplectic property.
- Prove that hyperverlet is convergent; a core property of useful solvers.
- Prove that the hyperbolic tangent is $r$-finite implying the more powerful $\tanh$-activated SympNet is a universal approximation of symplectic transformations.
- Show empirically that hyperverlet outperform velocity Verlet, HyperEuler, and SympNet.

## Related Work

The philosophy of hypersolvers is that by augmenting a solver of ODEs with a neural network, we can either achieve higher accuracy or faster inference (Shen, Cheng, and Liang 2020; Poli et al. 2020) - the trade-off between the two is the step size. For hypersolvers, a popular approach is to augment a traditional numerical solver with neural network (Shen, Cheng, and Liang 2020; Poli et al. 2020; Zhao et al. 2021). Other approaches include directly predicting the resulting value at a variable time (Mattheakis, Joy, and Protopapas 2021; Chen et al. 2020a) and architectures with closed-form time propagation (Hasani et al. 2021). We take the approach of augmenting a traditional solver, but only focus on Hamiltonian systems.

State of the art numerical solvers for Hamiltonian systems such as velocity Verlet (Swope et al. 1982), and Forest-Ruth (Forest and Ruth 1990) rely on symplecticity of the phase space flow. Alternatively to traditional solvers is symplectic neural networks. The methods invented integrate the symplecticity in two ways: replacing an agnostic solver with a symplectic solver like velocity Verlet (Chen et al. 2020b) or architectural choices ensuring the temporal evolution remains a symplectic transformation (Tong et al. 2021; Zhong, Dey, and Chakraborty 2020; Saemundsson et al. 2020; Jin et al. 2020). Interestingly, some of these solvers are non-convergent meaning that as step size of the discretization approaches zero, the solution does not converge to the exact solution, which is otherwise a core property of solvers.

## Preliminaries

We briefly introduce Hamiltonian mechanics and SympNets.

### Hamiltonian Mechanics

The class of systems for which our method applies is Hamiltonian systems, i.e. systems governed by a particular set of ordinary differential equations (ODE) called Hamilton's equations (Taylor 2005).

**Definition 1** (Hamiltonian system)**.** *A Hamiltonian system is defined by its energy function $H(q,p) = T(q,p) + V(q)$, the sum of kinetic and potential energy where $q, p \in \mathbb{R}^d$ denote the position and generalized momentum respectively. It is governed by the following two equations.*

$$\dot{q} = \frac{dq}{dt} = \frac{\partial H}{\partial p} \quad , \qquad \dot{p} = \frac{dp}{dt} = -\frac{\partial H}{\partial q} \tag{1}$$

*where $\dot{x}$ denotes the time derivative of $x$.*

To denote the system as a single differential equation, we let $z = (q, p)$. Then

$$\dot{z} = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \frac{\partial H}{\partial z} \tag{2}$$

**Definition 2** (Separable Hamiltonian system)**.** *A separable Hamiltonian system is the ordinary differential equation in Eq. 2 where the kinetic energy is only a function of the momentum, that is*

$$H(q,p) = T(p) + V(q) \tag{3}$$

A core property of Hamiltonian systems is that the phase flow is symplectic, a property of area-preservation.

**Definition 3** (Symplecticity)**.** *The phase flow of a Hamiltonian system defined by $H$ is symplectic. That is*

$$\nabla \cdot \dot{z} = 0 \tag{4}$$

Symplecticity is important for energy conservation, hence important for numerical solvers of Hamiltonian systems.

### Numerical Solvers

While desirable, analytical solutions to instantiations of Hamilton's equations often does not exist with proofs relying on Liouville's theorem and differential Galois theory. Instead, we rely on numerical approximations, which discretize the temporal axis into steps of equal size $h = t_{n+1} - t_n$ for all $n \in \mathbb{N}$. The oldest and simplest numerical solver is the Euler method.

$$\hat{z}_{n+1} = z_n + h \cdot f(z_n, t_n) \tag{5}$$

where $z_n = (q_n, p_n)$ is the state and $f = \dot{z}$ is the time derivative of the state. Euler method is convergent meaning that the local truncation error approaches zero as $h \to 0$.

**Definition 4** (Local truncation error)**.** *For a single-step method*

$$\Phi(z_n, t_n) = z_n + h \cdot \psi(z_n, t_n) \tag{6}$$

*where $\psi$ is the update function, the local truncation error is*

$$\tau_n = z_n - \Phi(z_{n-1}, t_{n-1}) \tag{7}$$

The update function $\psi$ for the Euler solver is $f$ but for more complex solvers, it can involve substeps. When the local truncation error of a solver is proportional to $h^{p+1}$, we call it a $p$th-order solver.

The Euler solver is unstable meaning that for bounded systems and step sizes $h$ larger than a threshold, the simulation is asymptotically unbounded. For Hamiltonian systems, we can exploit their symplecticity to simulate stable long-term trajectories (Hairer, Lubich, and Wanner 2003). A symplectic solver has a bounded error of energy.

A prevalent symplectic solver is velocity Verlet (Swope et al. 1982), which assumes the Hamiltonian is separable and exploit this to split the update of position and momentum.

**Definition 5** (Velocity Verlet)**.** *For a separable Hamiltonian system defined by $H$ with $\dot{p} = ma$ where $m$ is the mass and $a$ is the acceleration, velocity Verlet $\Phi_V$ is*

$$q_{n+1} = q_n + v_n \cdot h + \frac{1}{2} a(q_n) \cdot h^2 \tag{8}$$

$$v_{n+1} = v_n + \frac{a(q_n) + a(q_{n+1})}{2} \cdot h \tag{9}$$

## SympNet

SympNet is a set of symplectic neural network modules that provably can approximate any symplectic transformation, relying on the property that the composition of symplectic transformations is also a symplectic transformation. (Jin et al. 2020) present LA-SympNet, comprised of linear and activation modules, which satisfies a universal approximation theorem for symplectic transformations.

The linear layer $\Phi_L$ is computed as follows with two variants, upper $u$ and lower $l$ where they differ by the position of $S \cdot h$ and 0.

$$\Phi_L^u(z, h) = \begin{pmatrix} I & S \cdot h \\ 0 & I \end{pmatrix} z + h \cdot b \tag{10}$$

where $S \in \mathbb{R}^{d \times d}$ is a symmetric matrix, $I$ is the identity matrix, and $b \in \mathbb{R}^{2d}$ is the bias. To encoding the symmetry into the learning procedure, $S$ is computed as $S = A + A^T$ where $A$ is a $d \times d$ constraint-free parameter matrix.

The other layer to LA-SympNet is the activation layer, which is computed as

$$\Phi_A^u(z, h) = \begin{pmatrix} q \\ p \end{pmatrix} + \begin{pmatrix} a \odot \sigma(p) \cdot h \\ 0 \end{pmatrix} \tag{11}$$

where $a \in \mathbb{R}^d$, $\odot$ is the elementwise product, and $\sigma$ is an activation function. Similarly to the linear layer, it has a lower variant $l$ where the blocks of last vector of Eq. 11 are swapped and the activation is performed on $p$. The symplecticity of this layer can be proven by interpreting the additive update as an energy gradient. Composing multiple linear and activation layers conserves the symplecticity, and due to the block-triangular shape of the transformations, LA-SympNet is reversible.

## Problem Formulation

Assume a separable Hamiltonian system defined by $H$ and an initial condition $z_0 = (q_0, p_0)$. The goal for any numerical solver $\Phi$ is to solve the equation $z(t)$ for all time points $t_n = h \cdot n, 0 \leq n < N$ with minimal global truncation error. Assume additionally, a dataset $\mathcal{D}$ of trajectories of the system. For numerical hypersolvers with parameters $\theta$, the goal is to find the parameters for which the global truncation error is minimized.

$$\theta^* = \arg\min_\theta \sum_{n=1}^{N} z(t_n) - \Phi^n(z(t_0), t_0, h; \theta, \mathcal{D}) \tag{12}$$

where $\Phi^n$ denotes the application of $\Phi$ repeated $n$ times.

# Symplectic Hyperverlet

The architecture of hyperverlet is a predictor-corrector structure with the predictor being a traditional solver, namely velocity Verlet, and the corrector being a neural network. Velocity Verlet is a powerful symplectic solver prevalent in molecular dynamics. The choice of velocity Verlet restricts our method to separable Hamiltonian systems, which excludes systems like double pendulum or cartpole where the kinetic energy is a function of both position and momentum, but includes a wide range of systems like pendulum, spring-mass, molecular dynamics, and cosmic systems. We choose velocity Verlet despite this restriction for its unparalleled numerical stability and its simplicity.

While velocity Verlet is a numerically stable solver, errors can accumulate; especially phase errors for oscillating systems. A SympNet-based neural corrector can compensate for these errors yielding a higher accuracy while maintaining the symplecticity to improve long-term behavior.

To further improve the accuracy of the method, we propose three modifications to SympNet: parameterized transformations, hyperbolic tangent activation function, and replacing the step size $h$ with $h^3$.

SympNet is for systems identification, and not to generalize beyond a single set of time invariant properties $x$ such as mass, pendulum length, spring constant. To generalize, we exploit that the symplectic properties of SympNet is only dependent on the symmetry of the matrix $S$ and parameterize transformations with a neural network of $x$. That is, we replace the weight matrix $A$ in the linear layer of SympNet with $A(x; \theta_A)$ where $\theta_A$ are the parameters of a neural network. Similarly, we replace the bias $b$ with $b(x; \theta_b)$ and the weight vector $a$ of the activation layer with $a(x; \theta_a)$. The significance of this modification is observed in Fig. 2b where SympNet fails to capture energy conservation.

Next, we use the hyperbolic tangent as the activation function of the SympNet activation layer because it empirically improves the accuracy. However, (Jin et al. 2020) only prove that logistic sigmoid allows SympNet to approximate any symplectic transformation. We prove that hyperbolic tangent exhibits the same desirable traits.

The replacement of $h$ by $h^n, n \geq 2$ is necessary to prove the convergence of the hyperverlet, see Thm. 1. We choose $n = 3$ because velocity Verlet is a 2nd-order method meaning that its local truncation error is $\tau_n = O(h^3)$, hence the SympNet must correct an error proportional to $h^3$.

Combing the three modifications, Eq. 10 and 11 become

the following with $S = A(x;\theta_A) + A(x;\theta_A)^T$

$$\Phi_L^{u'}(z,h) = \begin{pmatrix} I & S \cdot h^3 \\ 0 & I \end{pmatrix} z + h^3 \cdot b(x,\theta_b) \tag{13}$$

$$\Phi_A^{u'}(z,h) = \begin{pmatrix} p \\ q \end{pmatrix} + \begin{pmatrix} a(x;\theta_a) \odot \tanh(q) \cdot h^3 \\ 0 \end{pmatrix} \tag{14}$$

Let $\Phi'_{LA}$ denote any LA-SympNet with our three modifications and $\Phi_V$ is velocity Verlet. Then hyperverlet is

$$\Phi_H(z,h) = \Phi'_{LA}(\Phi_V(z,h),h) \tag{15}$$

The design philosophy of SympNet rely on the fact that the set of symplectic transformations is a monoid with respect to composition to preserve the symplectic property of their architecture (Jin et al. 2020). This implies that hyperverlet is also a symplectic transformation.

While this informal proof is immediately obvious, other properties not so much. Namely, convergence is desirable, but non-obvious to prove. We also want to safely apply hyperbolic tangent as the activation function, maintaining the universal approximation property, which is conditioned on the function being $r$-finite. We prove the solver properties and $r$-finiteness of $\tanh$ in the next section.

## Theoretic Results

Convergence of a solver is the property that the numerical solution approaches the exact solution as $h \to 0$ and thus is a desirable trait. Similarly, the order of the solver is desirable to know, but since we do not predict the residual, we cannot employ the proof technique of (Poli et al. 2020). However, by applying Picard's theorem stating that if a solver is continuous in both $z$ and $h$, consistent, and Lipschitz continuous in $z$, it is convergent (Süli 2003), we proceed to prove that hyperverlet is convergent. Before proving the convergence of hyperverlet, we need an intermediate result characterizing Lipschitz continuity in the following.

**Lemma 1.** *The composition $f(x) = g(h(x))$ of two Lipschitz continuous functions $g, h$ with constants $K_g, K_h$ is also Lipschitz continuous with constant $K_f = K_g \cdot K_h$.*

*Proof.* Assume two Lipschitz continuous functions $g, h$ with constant $K_g, K_h$. By the definition of Lipschitz continuity, for any two $x_1, x_2$ we have

$$\begin{aligned} \|g(h(x_1)) - g(h(x_2))\| &\leq K_h \|h(x_1) - h(x_2)\| \\ &\leq K_h \cdot K_g \|x_1 - x_2\| \end{aligned} \tag{16}$$

Thus Lipschitz continuity is preserved under composition. □

**Theorem 1.** *For an initial value problem satisfying the conditions of Picard's Theorem and a Lipschitz continuous activation function $\sigma$, hyperverlet is convergent.*

*Proof.* If a single-step solver $\Phi(z,h)$ with an update function $\psi(z,h)$ is continuous in $z$ and $h$, consistent, and Lipschitz continuous in $z$, it is convergent (Süli 2003). Thus we prove each of the three conditions.

***Continuous*** Since the initial value problem is continuous by the conditions of Picard's Theorem, the acceleration function in Eq. 8 and 9 is continuous. Therefore, velocity Verlet is continuous.

For LA-SympNet, we prove the property for each module type. The linear layer is by definition of linearity everywhere-continuous. The activation layer is linear in the weight vector $a$ and the step size $h$. By the conditions of the theorem, the activation function is Lipschitz continuous, and by extension continuous. Finally, we observe that $h^3$ is continuous in $h$.

***Consistent*** Consistency of a method is if the local truncation error $\tau_n$ divided by the step size $h$ converges to zero as $h \to 0$.

$$\lim_{h \to 0} \frac{\tau_n}{h} = 0 \tag{17}$$

To prove consistency, first observe for Eq. 8 and 9 that the update function divided by $h$ is

$$\frac{\phi_q(q_{n-1},h)}{h} = v_{n-1} + \frac{1}{2} a(q_{n-1}) h \tag{18}$$

$$\frac{\phi_v(v_{n-1},h)}{h} = \frac{a(q_{n-1}) + a(q_n)}{2} \tag{19}$$

As $h$ approaches zero, this becomes $\dot{z} = (v, a)$ where $v$ denotes velocity and $a$ acceleration, which is exactly the function $f(z,t)$ or the right-hand side of the ODE.

For LA-SympNets with $h^3$ observe that for $h = 0$, both the linear and activation layer reduces to the identity transformation. Hence, hyperverlet is consistent.

***Lipschitz continuous*** The composition of two Lipschitz-continuous functions is also Lipschitz continuous and velocity Verlet is Lipschitz continuous (Madsen and Mathiesen 2021). Therefore, it suffices to show that LA-SympNets is Lipschitz continuous, which we may do by module due to Lemma 1. Starting with the linear layer, let

$$\psi_L^u(z,h) = \begin{pmatrix} 0 & S \cdot h^3 \\ 0 & 0 \end{pmatrix} z + h^3 \cdot b \tag{20}$$

Then $\psi_L^u$ is Lipschitz continuous in $z$ if

$$\|\psi_L^u(z_1,h) - \psi_L^u(z_2,h)\| \leq K \|z_1 - z_2\| \tag{21}$$

For this particular operation, we may compute the Lipschitz constant directly $K = h^3 \max |S|$. The proof holds for a lower linear layer too. The update function of the upper activation layer is $(a \odot \sigma(q) \cdot h^3, 0)$. By Lemma 1, the Lipschitz continuity of linearity, and the condition of the theorem that $\sigma$ is Lipschitz continuous, the activation layer is Lipschitz continuous, and again the proof holds for the lower variant.

Therefore, hyperverlet is convergent. □

(Jin et al. 2020) prove that LA-SympNets is a universal approximation for all symplectic transformations if the activation function is $r$-finite, and prove that logistic sigmoid $\sigma$ exhibit this property. An alternative sigmoid-like function that is desirable for its higher convergence rate is the hyperbolic tangent function $\tanh$ (Dreyfus 2005). Here, we prove that $\tanh$ is $r$-finite for any non-negative integer $r$. Starting with the definition of $r$-finiteness (Jin et al. 2020).

**Definition 6** ($r$-finiteness)**.** *Let $\mathbb{N}^0 = \{0\} \cup \mathbb{N}$ where $\mathbb{N}$ is the set of positive integers. A function $f$ is $r$-finite for some $r \in \mathbb{N}^0$ if $f$ is a smooth function satisfying*

$$0 < \int \left| \frac{d^r}{dx^r} f \right| d\lambda < \infty \tag{22}$$

*where $\lambda$ denotes a Lebesgue measure on $\mathbb{R}$.*

To prove that $\tanh$ is $r$-finite, we follow the same proof structure as Lemma 1 of (Jin et al. 2020).

**Theorem 2.** *The hyperbolic tangent function $\tanh$ is $r$-finite for any $r \in \mathbb{N}^0$.*

*Proof.* The derivative of the hyperbolic tangent is bounded $0 < \tanh'(x) \leq 1$ for all $x$ so

$$\int |\tanh'(x)| d\lambda = \int \tanh'(x) d\lambda = 2 \tag{23}$$

By mathematical induction, we may easily show by exploiting that $\tanh'(x) = 1 - \tanh^2(x)$ that for $n \geq 2$

$$\tanh^{(n)}(x) = \tanh'(x) P^{(n-1)}(\tanh(x)) \tag{24}$$

where $P^{(n-1)}$ denotes an $n-1$ order polynomial. Then

$$\begin{aligned} 0 &< \int |\tanh^{(r)}| d\lambda \\ &\leq \int |\tanh'(x)| d\lambda \cdot \left( \sup |P^{(r-1)}(\tanh(x))| \right) \\ &= \sup_{y \in (-1,1)} |P^{(r-1)}(y)| < \infty \end{aligned} \tag{25}$$

Therefore $\tanh$ is $r$-finite for any $r \in \mathbb{N}^0$. □

With this property, we can safely apply $\tanh$ as the activation function while maintaining the universal approximation property, which improves the accuracy of hyperverlet.

## Experiment Setup

We test the performance on the two classical systems, undamped spring-mass and pendulum (Greydanus, Dzamba, and Yosinski 2019). They are chosen because they are simple separable Hamiltonian systems, which allows comparing solvers. To synthesize, the training and test data, we utilize a 4th-order Forest-Ruth symplectic solver (Forest and Ruth 1990) with small time steps to produce a high-precision dataset, which we coarsen by an integer factor to obtain the final dataset.

### Spring-mass

The simplest of the studied systems is the spring-mass where a positive point-mass $m$ is attached to a Hookian spring, which oscillates back and forth indefinitely. We measure the position $q$ of the point-mass relative to the mount of the spring, and denote the idle extension $l$. A force is applied towards the idle extension linearly by a spring constant $k$. The Hamiltonian dynamics of system are

$$\dot{q} = \frac{p}{m} \quad , \qquad \dot{p} = -k(q - l) \tag{26}$$

| **Parameter** | Spring-mass | Pendulum |
|---|---|---|
| Step size $h$ | $\mathcal{N}(1\text{e}{-3}, 1\text{e}{-4})$ | $\mathcal{N}(1\text{e}{-3}, 1\text{e}{-4})$ |
| Number of steps | 100000 | 600000 |
| Length $l$ | $\mathcal{U}(0.5, 2.0)$ | $\mathcal{U}(0.5, 2.0)$ |
| Spring constant $k$ | $\mathcal{U}(0.8, 1.2)$ | - |
| Mass $m$ | $\mathcal{U}(0.9, 1.1)$ | $\mathcal{U}(0.9, 1.1)$ |

Table 2: Parameters of the dataset generation using a higher-order solver with small time steps. $\mathcal{N}(\mu, \sigma)$ and $\mathcal{U}(l, u)$ denotes a normal and uniform distribution.

Note that there exists an analytical solution to this system (Madsen and Mathiesen 2021). The time invariant properties $x$ for the spring-mass system are the mass $m$, spring constant $k$, and idle length $l$. We generate this dataset by drawing from the arbitrarily chosen distributions listed in Table 2.

### Pendulum

The frictionless pendulum distinguishes from the spring-mass system by polar coordinates instead of Cartesian and non-linear equations of motion. The choice of polar coordinates is natural as the position and momentum along the magnitude axis is constant and thus ignorable. The position on the magnitude axis is the length of the pendulum $l$, and the driving force is gravity with acceleration of $g$. The dynamics of system are

$$\dot{q} = \frac{p}{ml^2} \quad , \qquad \dot{p} = -mgl \sin q \tag{27}$$

Note that the momentum is not $p = mv$ as usual, which is due to the polar coordinates. The time invariant properties $x$ for the pendulum system are the mass $m$ and pendulum length $l$. We generate this dataset by drawing from the arbitrarily chosen distributions listed in Table 2.

### Baselines

As baselines, we employ the following four solvers:

- **Euler** - a non-symplectic solver that is simple but notoriously unstable,
- **HyperEuler** (Poli et al. 2020) - a non-symplectic hypersolver that improves the accuracy of the Euler solver by predicting the local truncation error,
- **Velocity Verlet** - a symplectic solver used for separable Hamiltonian systems, and
- **SympNet** (Jin et al. 2020) - a symplectic neural network that is non-convergent.

We only consider up to 2-order solvers because the goal is to accelerate computations for systems with a computationally heavy acceleration evaluation. Hence we exclude solvers with more evaluations, such as 4-order solvers RK4 (Süli 2003) and FR4 (Forest and Ruth 1990).

### Training Procedure and Hyperparameters

For the neural network of HyperEuler, we use a fully connected neural network with 4 hidden layers of 16 neurons, sigmoid activation, and no activation on the output layer.

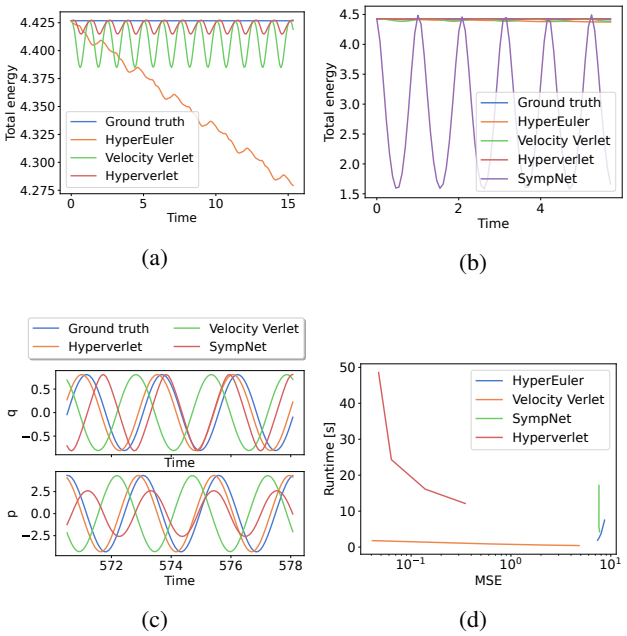


Figure 2: Results of the solver comparison by energy conservation and phase errors. **(a)** shows the total energy as a function of time for the pendulum system comparing hyperverlet against 1st- and 2nd-order baselines. Euler is excluded for its large energy accumulation. **(b)** shows a subset of (a) with SympNet included to observe its large energy bound. **(c)** shows the position and momentum in a window of $[7400, 7500]$ time steps for a single trial to observe phase errors, namely that velocity Verlet desynchronizes with the ground truth. **(d)** shows a comparison of runtimes as a function accuracy. Euler is excluded for its large loss. Note that the number of samples for the runtime study is very low, and that implementations were not optimized for speed.

The weights are initialized using kaiming normal initialization, and bias is initialized to all zeros. The SympNet architecture consists of 4 alternating linear layers starting with an upper module, followed by one lower activation layer with $\tanh$ as the activation function. The structure is repeated with 4 linear layers and a upper activation layer. This 10 layer structure is repeated twice. The corrector for hyperverlet is the same architecture except for the modifications detailed in Sec.  where the parameterization is a fully connected network with a single hidden layer of 16 neurons. We adopt the initialization procedure of SympNet (Jin et al. 2020) where the weights are initialized randomly from the distribution $\mathcal{N}(0, 0.01)$, and bias is initialized to all zeros.

For all trainable solvers, we employ a learning rate of $1\mathrm{e}{-3}$, an Adam optimizer with weight decay of $1\mathrm{e}{-2}$, and the loss is MSE of single-step predictions. Solvers are trained for $6c$ epochs where $c$ denotes the integer coarsening factor relative to the high precision dataset. The dependence on the coarsening factor is due to the less amount of data available for higher coarsening. We train all solvers using a batch size of $100{,}000$.

## Implementation Details

Experiments are conducted on a Linux Manjaro desktop with an Intel i7-6700k processor and 16GB RAM. No GPU is used for the reported experiments. The method is implemented in Pytorch 1.9.0, and the code is available at https://github.com/Zinoex/hyperverlet. All randomness is seeded in Pytorch, Numpy, and Python arbitrarily with 42.

## Experiment Results

To measure the accuracy, we employ the traditional mean squared error (MSE) of the canonical coordinates $z$, i.e. both position and momentum, where the average is computed over the temporal and spatial axis. We report the mean and standard deviation of the MSE over trials as $\mu \pm \sigma$ in Table 3 for both the spring-mass and pendulum systems. The number of trials are 100 for both systems.

Comparing Euler and HyperEuler in Table 3, we see the accuracy rapidly decreases as the step size $h$ increases, which is a result of its instability and lack of energy conservation. Hyperverlet outperforms both testifying to the importance of the base solver. We also find that the accuracy of hyperverlet scales better with higher step sizes compared to velocity Verlet. Finally, we see that SympNet is largely unaffected by step size, but this is because it fails at capturing energy conservation and by extension time evolution.

Analysis of energy conservation is shown in Fig. 2a and 2b. Note the figures show a single trial so the results may not generalize, but still reveal very interesting properties of each solver. Observing the symplectic solvers, i.e. velocity Verlet, SympNet, and hyperverlet, we find that bounded oscillations rather than energy conservation is inherent to symplecticity. SympNet is also only listed in short-term energy plot in Fig. 2b because its oscillation bounds are enormous and would be hiding the details of the other solvers.

To analyze phase errors, we analyze the phase space as shown in Fig. 2c where the window of $[7400, 7500]$ time steps corresponding to the last $100$ samples from one trajectory of the pendulum system. We find that velocity Verlet is out of sync whereas hyperverlet is almost synchronous with the ground truth. We also observe that SympNet predicts a low amplitude of the oscillations in momentum.

Figure 2d shows the runtime as a function of accuracy. Note that for each coarsening factor and solver pair, there is only one sample, and that implementations were not optimized for speed.

## Ablation Study

To study the impact of a symplectic corrector, the parameterization of the symplectic transformations, and the choice of activation function, we conduct an ablation study. The experiments are conducted on the pendulum system with mean step size $h = 0.08$ as it is the most difficult case studied.

We study the cases listed in Table 4. The first case is a non-symplectic corrector, i.e. a neural network that predicts the local truncation error similar to HyperEuler. The neural network is the same structure as for HyperEuler.

The other 4 cases are all symplectic where we study our modifications. As a baseline, we use SympNet as a correc-

| **Spring-mass** | Mean step size $h$ | | | |
|---|---|---|---|---|
| Solver | 0.025 | 0.05 | 0.1 | 0.2 |
| Euler | $2.4\text{e}{+14} \pm 1.7\text{e}{+15}$ | - | - | - |
| HyperEuler | $1.0\text{e}{-3} \pm 3.4\text{e}{-3}$ | $3.0\text{e}{-3} \pm 7.1\text{e}{-3}$ | $5.6\text{e}{-1} \pm 5.5\text{e}{+0}$ | - |
| Velocity Verlet | $7.7\text{e}{-5} \pm 1.2\text{e}{-4}$ | $1.2\text{e}{-3} \pm 1.8\text{e}{-3}$ | $1.9\text{e}{-2} \pm 2.9\text{e}{-2}$ | $2.5\text{e}{-1} \pm 3.6\text{e}{-1}$ |
| SympNet | $6.9\text{e}{-1} \pm 7.8\text{e}{-1}$ | $6.9\text{e}{-1} \pm 7.7\text{e}{-1}$ | $6.9\text{e}{-1} \pm 7.7\text{e}{-1}$ | $6.9\text{e}{-1} \pm 7.6\text{e}{-1}$ |
| Hyperverlet | $\mathbf{1.2\text{e}{-5} \pm 3.8\text{e}{-5}}$ | $\mathbf{2.2\text{e}{-5} \pm 4.9\text{e}{-5}}$ | $\mathbf{4.7\text{e}{-5} \pm 1.6\text{e}{-4}}$ | $\mathbf{7.6\text{e}{-4} \pm 2.2\text{e}{-3}}$ |
| **Pendulum** | Mean step size $h$ | | | |
| Solver | 0.02 | 0.04 | 0.06 | 0.08 |
| Euler | $2.1\text{e}{+7} \pm 2.5\text{e}{+7}$ | $3.8\text{e}{+7} \pm 3.8\text{e}{+7}$ | $4.3\text{e}{+7} \pm 3.0\text{e}{+7}$ | $4.1\text{e}{+7} \pm 1.9\text{e}{+7}$ |
| HyperEuler | $8.7\text{e}{+0} \pm 1.2\text{e}{+1}$ | $8.0\text{e}{+0} \pm 1.1\text{e}{+1}$ | $7.6\text{e}{+0} \pm 1.1\text{e}{+1}$ | $7.4\text{e}{+0} \pm 9.5\text{e}{+0}$ |
| Velocity Verlet | $\mathbf{4.1\text{e}{-2} \pm 4.4\text{e}{-2}}$ | $5.8\text{e}{-1} \pm 5.7\text{e}{-1}$ | $2.2\text{e}{+0} \pm 2.1\text{e}{+0}$ | $4.8\text{e}{+0} \pm 5.1\text{e}{+0}$ |
| SympNet | $7.6\text{e}{+0} \pm 9.0\text{e}{+0}$ | $7.6\text{e}{+0} \pm 9.0\text{e}{+0}$ | $7.6\text{e}{+0} \pm 9.0\text{e}{+0}$ | $7.7\text{e}{+0} \pm 9.2\text{e}{+0}$ |
| Hyperverlet | $4.7\text{e}{-2} \pm 9.2\text{e}{-2}$ | $\mathbf{6.3\text{e}{-2} \pm 1.5\text{e}{-1}}$ | $\mathbf{1.4\text{e}{-1} \pm 2.6\text{e}{-1}}$ | $\mathbf{3.5\text{e}{-1} \pm 5.2\text{e}{-1}}$ |

Table 3: MSE of the canonical coordinates of each solver for both the spring-mass and pendulum systems. $\mu \pm \sigma$ denotes the mean and standard deviation of the MSE over 100 trials. The symbol - means that the MSE was $inf$, which is due to the instability of Euler. Bold denotes the best MSE for each system and step size.

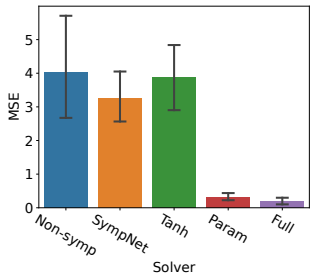


Figure 3: Results of the ablation study. The whiskers denote the 95% confidence interval. The meaning of the solver names is listed in Table 4.

| Name | Sympletic | Parameterized | Activation |
|---|---|---|---|
| Non-symp | | - | - |
| SympNet | ✓ | | Sigmoid |
| Tanh | ✓ | | Tanh |
| Param | ✓ | ✓ | Sigmoid |
| Full | ✓ | ✓ | Tanh |

Table 4: Model properties in the ablation study.

tor and substitute logistic sigmoid with hyperbolic tangent, static weights for dynamics weights, or both. The parameterizaton of the weights are the same as for hyperverlet.

The results of the ablation study are shown in Fig. 3. While symplecticity alone does not yield significant improvement, combined with parameterized transformations it greatly improves accuracy. Regarding the activation function, we interestingly observe that hyperbolic tangent only improves accuracy when SympNet is parameterized. This improvement is significant with a $42\%$ decrease in mean MSE over the 100 trials.

## Conclusions and Outlook

We present Hyperverlet, a novel solver combining velocity Verlet, a traditional, symplectic solver for separable Hamiltonian systems, with SympNet, a symplectic neural network, to achieve state-of-the-art results on a spring-mass and pendulum system. We modify the SympNet architecture with parameterization and $\tanh$ activation to improve the accuracy of our method, and prove that the solver is convergent. Hyperverlet is symplectic, i.e. preserving area of the phase flow, which is desired for simulating Hamiltonian systems. As a final theoretic result, we show that hyperbolic tangent is $r$-finite, which is a necessary condition for SympNet to be a universal approximation of symplectic transformations.

Similarly to (Jin et al. 2020), we believe that exploiting the underlying geometrical structures are vital to improving the accuracy of simulations. An interesting direction is the extension to graph neural networks. For systems like molecular dynamics and cosmic models where the number of bodies model may vary, the scalability is interesting as it allows reusing the solver with retraining.

An important limitation that hyperverlet can only solve problems with separable Hamiltonian systems, meaning that the kinetic energy is only a function of the momentum. This excludes systems like the double pendulum or the autonomous cartpole system, but includes a wide range of systems like the pendulum, molecular dynamics, and cosmic models. Future research may focus on how to apply symplectic hypersolvers for non-separable Hamiltonian systems.

## Acknowledgments

We would like to extend our sincere thanks to Anders Madsen for invaluable contribution as part of his Master thesis. Thanks also to Nicklas Hansen for feedback on the paper, and to Nikolaj Jensen Ulrik and Simon Virenfeldt for useful discussions throughout the project.

This work was partially supported by Independent Research Fund Denmark under agreements 8022-00246B and 8048-00038B.

## References

Chen, F.; Sondak, D.; Protopapas, P.; Mattheakis, M.; Liu, S.; Agarwal, D.; and Giovanni, M. D. 2020a. NeuroDiffEq: A Python package for solving differential equations with neural networks. *Journal of Open Source Software*, 5(46): 1931.

Chen, Z.; Zhang, J.; Arjovsky, M.; and Bottou, L. 2020b. Symplectic Recurrent Neural Networks. In *International Conference on Learning Representations*.

Dreyfus, G. 2005. *Neural Networks*. Springer-Verlag.

Forest, E.; and Ruth, R. D. 1990. Fourth-order symplectic integration. *Physica D: Nonlinear Phenomena*.

Greydanus, S.; Dzamba, M.; and Yosinski, J. 2019. Hamiltonian Neural Networks. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Hairer, E.; Lubich, C.; and Wanner, G. 2003. Geometric numerical integration illustrated by the Störmer–Verlet method. *Acta Numerica*, 12: 399–450.

Hasani, R.; Lechner, M.; Amini, A.; Liebenwein, L.; Tschaikowski, M.; Teschl, G.; and Rus, D. 2021. Closed-form Continuous-Depth Models. arXiv:2106.13898.

Jin, P.; Zhang, Z.; Zhu, A.; Tang, Y.; and Karniadakis, G. E. 2020. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, 132: 166–179.

Madsen, A.; and Mathiesen, F. B. 2021. *HyperVerlet: A Deep Learning Method for Numerically Solving Initial Value Problems of Hamiltonian Systems*. Master's thesis, Aalborg University.

Mattheakis, M.; Joy, H.; and Protopapas, P. 2021. Unsupervised Reservoir Computing for Solving Ordinary Differential Equations. arXiv:2108.11417.

Poli, M.; Massaroli, S.; Yamashita, A.; Asama, H.; and Park, J. 2020. Hypersolvers: Toward Fast Continuous-Depth Models. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 21105–21117. Curran Associates, Inc.

Saemundsson, S.; Terenin, A.; Hofmann, K.; and Deisenroth, M. 2020. Variational Integrator Networks for Physically Structured Embeddings. In Chiappa, S.; and Calandra, R., eds., *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, 3078–3087. PMLR.

Shen, X.; Cheng, X.; and Liang, K. 2020. Deep Euler method: solving ODEs by approximating the local truncation error of the Euler method. arXiv:2003.09573.

Süli, E. 2003. *An introduction to numerical analysis*. Cambridge University Press. ISBN 1-107-13229-0.

Swope, W. C.; Andersen, H. C.; Berens, P. H.; and Wilson, K. R. 1982. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *The Journal of Chemical Physics*, 76(1): 637–649.

Taylor, J. R. 2005. *Classical mechanics*. University Science Books.

Tong, Y.; Xiong, S.; He, X.; Pan, G.; and Zhu, B. 2021. Symplectic neural networks in Taylor series form for Hamiltonian systems. *Journal of Computational Physics*, 437: 110325.

Zhao, F.; Chen, X.; Wang, J.; Shi, Z.; and Huang, S.-L. 2021. Performance-Guaranteed ODE Solvers with Complexity-Informed Neural Networks. In *The Symbiosis of Deep Learning and Differential Equations*.

Zhong, Y. D.; Dey, B.; and Chakraborty, A. 2020. Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control. In *International Conference on Learning Representations*.