Neural Symplectic Integrator with Hamiltonian Inductive Bias for the Gravitational N-body Problem

Maxwell X. Cai
SURF Corporative / Leiden University
Science Park 140, 1098 XG Amsterdam / Niels Bohrweg 2, 2300 RA Leiden
The Netherlands
maxwell.cai@surf.nl

Simon Portegies Zwart
Leiden University
Niels Bohrweg 2, 2300 RA Leiden
spz@strw.leidenuniv.nl

Damian Podareanu
SURF Corporative
Science Park 140, 1098 XG Amsterdam
damian.podareanu@surf.nl

Abstract

The gravitational N-body problem, which is fundamentally important in astrophysics to predict the motion of N celestial bodies under the mutual gravity of each other, is usually solved numerically because there is no known general analytical solution for $N > 2$. Can an N-body problem be solved accurately by a neural network (NN)? Can a NN observe long-term conservation of energy and orbital angular momentum? Inspired by Wistom & Holman's symplectic map, we present a neural N-body integrator for splitting the Hamiltonian into a two-body part, solvable analytically, and an interaction part that we approximate with a NN. Our neural symplectic N-body code integrates a general three-body system for 10^5 steps without diverting from the ground truth dynamics obtained from a traditional N-body integrator. Moreover, it exhibits good inductive bias by successfully predicting the evolution of N-body systems that are no part of the training set.

1 Introduction

Deep neural networks (DNNs) have been widely used as universal function approximators to learn a target distribution. Such a property is promising for physics-related applications, whose underlying distributions are typically high-dimensional, computationally expensive, and/or difficult to derive analytically. In astrophysics and particle physics, generative models are often used for Monte-Carlo event generation [e.g., Erdmann et al., 2018, Zamudio-Fernandez et al., 2019], dimensionality reduction [e.g., Portillo et al., 2020], classification [e.g., Fremling et al., 2021], and anomaly detection [e.g., Reis et al., 2018].

Despite the success of the aforementioned DNN applications on astrophysical data, the trained models learn to encode representations from the training data, but they do not necessarily understand the underlying physical laws. Making DNNs understand physical laws, therefore, requires special formulations. Physics informed neural networks [PINNs, see Raissi et al., 2017a,b, Raissi et al., 2019], for example, take the form of differential equations and replace the differentiable mathematical function f (which is sometimes unknown or expensive to calculate) by a differentiable neural network $f_{\rm NN}$. In this way, PINNs are trained to approximate the function f, and thus differentiating the PINNs will yield the evolution of the underlying system. Moreover, the outputs of $f_{\rm NN}$ becomes more interpretable. This idea is echoed in the recent work by Greydanus et al. [2019], in which the authors replace the Hamiltonian of a dynamics system with a DNN and successfully apply it to solved

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

problems such as pendulum, mass-spring, and the two-body systems. The resulting architecture, namely Hamiltonian Neural Networks (HNNs), manages to learn and respect the conservation of total energy. Similarly, the Hamiltonian Generative Networks (HGNs) are capable of consistently learning Hamiltonian dynamics from high-dimensional observations (such as images) without restrictive domain assumptions, and the Neural Hamiltonian Flow (NHF), which uses Hamiltonian dynamics to model expressive densities [Toth et al., 2019]. Various improvements and adoptations have since been made, see e.g., Chen et al. [2019], Sanchez-Gonzalez et al. [2019], Cranmer et al. [2020], Desmond Zhong et al. [2019], Finzi et al. [2020].

All aforementioned works approximate the full Hamiltonian (or a general function f) using a neural network. In this paper, however, we propose a formulation to improve the approximation. We will see that, if the Hamiltonian can be rewritten as an analytically solvable part and a residual part, then approximating only the small residual part with a DNN will improve the prediction accuracy. As an application, we use this idea to construct a neural symplectic integrator for N-body simulations [1], based on the formulation by Wisdom and Holman [1991].

2 Methods

2.1 Theoretical Framework

Hamiltonian mechanics is a reformulation of Newtonian mechanics, defined as the sum of potential energy and kinetic energy, i.e., $\mathcal{H} = T + V$, where T is the generalized kinetic energy and V is the generalized potential energy. The function $\mathcal{H}$ is called the Hamiltonian of the system, defined in an abstract phase space $\mathbf{s} = (\mathbf{q}, \mathbf{p})$, where $\mathbf{q} = (q_1, q_2, ..., q_n)$ is the position vector, and $\mathbf{p} = (p_1, p_2, ..., p_n)$ is the momentum vector. The subscript n indicates the dimensionality of the coordinates. The dynamics can be obtained through the following set of differential equations:

$$\dot{\mathbf{q}} \equiv \frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \qquad \dot{\mathbf{p}} \equiv \frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}. \tag{1}$$

Since $\mathcal{H}$ is the total energy, it is a conserved quantity in systems where external perturbations do not exist or can be safely ignored. For a gravitational N-body systems, each celestial body experiences the gravity exerted by the other $N-1$ bodies, and the corresponding Hamiltonian is

$$\mathcal{H} = \sum_{i=0}^{N-1} \frac{p_i^2}{2m_i} - G \sum_{i=0}^{N-2} m_i \sum_{j=i+1}^{N-1} \frac{m_j}{r_{ij}}, \tag{2}$$

where $p_i \equiv |\mathbf{p}_i|$ is the magnitude of the momentum vector for particle i, and $r_{ij} \equiv |\mathbf{q}_i - \mathbf{q}_j|$ is the distance between particles i and j. While there is no known analytic solution for the N-body problems with $N > 2$, Wisdom and Holman [1991] point out that the Hamiltonian of an N-body problem can be naturally split into two parts:

$$\mathcal{H} = \mathcal{H}_{\text{kepler}} + \mathcal{H}_{\text{inter}}. \tag{3}$$

Here $\mathcal{H}_{\text{kepler}}$ represents the Kepler motion of the bodies with respect to the center of mass, and $\mathcal{H}_{\text{inter}}$ represents the perturbation of the minor bodies on each other. This splitting is rooted in the perturbation theory, and this formulation allows the integrator to reduce numerical errors by taking the advantage that $\mathcal{H}_{\text{kepler}}$ can be solved analytically. Working in Jacobi coordinates, $\mathcal{H}_{\text{kepler}}$ can be written as

$$\mathcal{H}_{\text{kepler}} = \sum_{i=1}^{N-1} \left(\frac{\tilde{p}_i^2}{2\tilde{m}_i} - G\frac{\tilde{M}_i \tilde{m}_i}{\tilde{r}_i} \right). \tag{4}$$

Here $\tilde{m}_i = \left(\sum_{j=0}^{i-1} m_j / \sum_{j=0}^{i} m_j \right) m_i$ is the normalized mass for body i, $\tilde{p}_i = \tilde{m}_i \tilde{v}_i$ is the momentum of body i in Jacobi coordinates, and $\tilde{r}_i$ is the position vector of body i in Jacobi coordinates. The interacting Hamiltonian $\mathcal{H}_{\text{inter}}$ takes the form

$$\mathcal{H}_{\text{inter}} = G \sum_{i=1}^{N-1} \frac{m_0 m_i}{\tilde{r}_i} - G \sum_{i=0}^{N-2} m_i \sum_{j=i+1}^{N-1} \frac{m_j}{r_{ij}}. \tag{5}$$

[1] Source code available at `https://github.com/maxwelltsai/neural-symplectic-integrator`.

The Wisdom-Holman (WH) integrator is a symplectic mapping that employs a kick-drift-kick (or drift-kick-drift) strategy. During the drift step, the integrator assumes that the secondary body orbits the central body (or the center of mass) without experiencing any perturbations. This assumption allows one to propagate the state analytically by solving Kepler's equation. The error for leaving the perturbations from other bodies unaccounted for is subsequently corrected by the kick step, in which the mutual interactions between each body (e.g., planets) are regarded as velocity kicks. A velocity kick $\Delta\mathbf{v} = \mathbf{a}_{\rm p}\Delta t$ accounts for the cumulative effect of all planets due to the perturbing acceleration $\mathbf{a}_{\rm p}$ over a time interval Δt. Since a kick step only changes the velocity vector without touching the position vector, it ensures that the evolution of position vectors is smooth and differentiable.

2.2 Neural Interacting Hamiltonian (NIH)

$\mathcal{H}_{\rm kepler}$ is an $\mathcal{O}(N)$ operation and $\mathcal{H}_{\rm inter}$ is a more expensive $\mathcal{O}(N^2)$ operation. The former can be solved analytically, but the latter has to be solved numerically. We solve $\mathcal{H}_{\rm kepler}$ analytically but $\mathcal{H}_{\rm inter}$ using a neural network. Following Greydanus et al. [2019], we use a simple multi-layer perceptron (MLP) backbone network $\mathcal{H}_{{\rm inter},\theta}$ to serve as a function approximator of $\mathcal{H}_{\rm inter}$. The network, which we refer to as the neural interacting Hamiltonian (NIH), outputs a single scalar value analog to the total energy of the interaction part. This network is differentiable, and by taking its in-graph gradients with respect to $\mathbf{q}$ and $\mathbf{p}$, we obtain the velocities and the accelerations, respectively. The accelerations are used to calculate the velocity kicks in the WH mapping. Therefore, the networks takes the state vector $(\mathbf{q}, \mathbf{p})$ of the underlying N-body system as an input, and generates the time derivative $(\dot{\mathbf{q}}_\theta, \dot{\mathbf{p}}_\theta)$ (every quantity related to the neural network is hereafter denoted with the subscript θ). The loss is calculated as

$$\mathcal{L} == \underbrace{\|\dot{\mathbf{q}} - \dot{\mathbf{q}}_\theta\|^2}_{L_2(v)} + \underbrace{\|\dot{\mathbf{p}} - \dot{\mathbf{p}}_\theta\|^2}_{L_2(a)} + \underbrace{\|\dot{\mathbf{q}}_\theta \times \mathbf{p} - \mathbf{q} \times \dot{\mathbf{p}}_\theta\|^2}_{\dot{\mathbf{L}}_{\rm residual}}. \tag{6}$$

Here $\dot{\mathbf{q}}$ and $\dot{\mathbf{p}}$ are ground truth velocities and accelerations obtained from a traditional WH integrator. The loss function consists of three parts: the first part $L_2(v)$ aims to minimize the difference between the predicted velocities and the ground truth velocities; the second part $L_2(a)$ aims to minimize the difference between the predicted accelerations and the ground truth acceleration. The third part $\dot{\mathbf{L}}_{\rm residual}$ is essentially the time derivative of the orbital angular momentum $\mathbf{L}$:

$$\frac{d\mathbf{L}}{dt} \equiv \frac{d\left(\mathbf{q}\times\mathbf{p}\right)}{dt} = \frac{d\mathbf{q}}{dt}\times\mathbf{p} + \mathbf{q}\times\frac{d\mathbf{p}}{dt} = \frac{\partial\mathcal{H}_\theta}{\partial\mathbf{p}}\times\mathbf{p} - \mathbf{q}\times\frac{\partial\mathcal{H}_\theta}{\partial\mathbf{q}} \tag{7}$$

In the absence of external perturbations, there is no external torque and the total angular momentum of an N-body system is conserved, hence $d\mathbf{L}/dt = 0$. Therefore, minimizing the residual of $d\mathbf{L}/dt$ is equivalent to forcing $\mathcal{H}_{{\rm inter},\theta}$ to obey the law of angular momentum conservation.

In an attempt to improve the backbone network, we also tested a sparsely-connected MLP model [see e.g., Constantin Mocanu et al., 2017, Pieterse and Constantin Mocanu, 2019] where 90% of its neurons are randomly zeroed out as a form of regularization. This approach forces a more efficient use of neurons. The sparse MLP model performs slightly better than the vanilla MLP backbone. For simplicity, hereafter we use only the vanilla MLP backbone to carry out all the tests.

2.3 Simulations, Activation Function, and Training:

The `SymmetricLog` Activation Function Since the neural N-body integrator is expected to handle real astrophysical data like any traditional integrators, we need to make sure that $\mathcal{H}_{{\rm inter},\theta}$ is capable of handling raw data without the need to perform data pre-processing. Given the vast dynamic ranges in astrophysical data, however, commonly used activation functions, such as `ReLU`, `sigmoid`, and `tanh` have insufficient dynamic range. For example, the `ReLU` activation function does not activate for negative values, and the `tanh` activation function saturates to 1 or -1 for absolute value > 3. As such, we design a custom activation function `SymmetricLog`:

$$f(x) = \tanh(x)\log\left[x\tanh(x) + 1\right]. \tag{8}$$

As shown in Fig. 1, the `SymmetricLog` activation function behaves similarly as `tanh` for values close to zero, which delivers strong gradients for backward propagation; for larger values, it behaves like a logarithmic function, which does not saturate or explode the gradients; it is well defined for

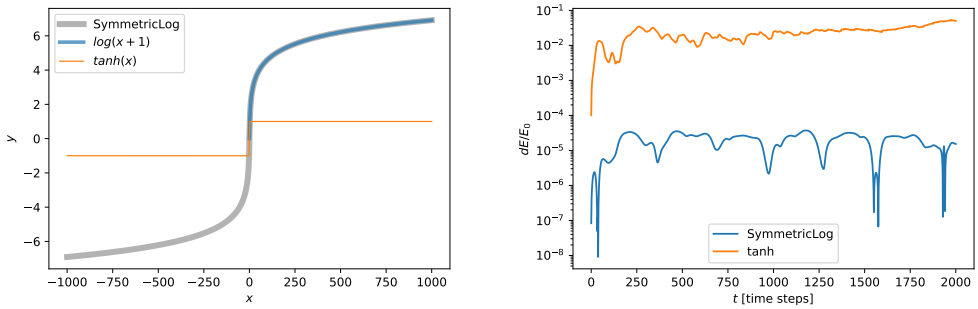

Figure 1: Left: The activation function used in the neural interaction Hamiltonian, as compared to $y = \log(x + 1)$ and $y = \tanh(x)$. Right: Relative integration energy error dE/E_0 as a function of time. The `SymmetricLog` activation performs ~ 3 orders of magnitude better in terms of energy conservation compared to `tanh`.

any real values, making it possible for the neural network to handle raw astrophysical data (unless the quantities are smaller than 10^{-3} or larger than 10^4, in which case data normalization is needed to achieve optimal prediction accuracy). Its symmetry with respect to zero makes it particularly suitable for using in N-body simulations because the coordinates of celestial bodies are usually quasi-symmetric with respect to the center of the reference frame. We perform a test of planetary systems with 10 planets orbiting a solar-type star. The semi-major axes of these planets vary from 0.5 au to 10 au, which means that any planets with a semi-major axis greater than ~ 3 au will saturate the `tanh` activation function. `ReLU` and `SoftPlus` cannot be used because they cannot generate negative activation outputs. As shown in Fig. 1, `SymmetricLog` performs ~ 3 orders of magnitude better in terms of energy conservation.

Training We deliberately train the NIH with a small number of simulations (see Appendix A.1 for the initial conditions). In doing so, we wanted to test whether the sub-optimally trained neural network has good inductive bias in transferring the physics laws it learn from the training examples to distinctively different N-body systems. We train the neural network with only $N = 3$ systems (there is no NIH for $N = 2$), but the *same* network is subsequently asked to deal with problems with arbitrary N. For brevity, we refer to a traditional Wisdom-Holman integrator as `WH`, and the modified Wisdom-Holman integrator with neural interacting Hamiltonian as `WH-NIH`. We implement `WH-NIH` using `PyTorch`, and use the `Adam` [Kingma and Ba, 2014] optimizer to train the NIH. Given that the training dataset is deliberately made small (roughly 300 MB of `float32` data), full training can be done on a single GPU or a multi-core CPU within one hour.

3 Results

3.1 Two-body Problems

We test the neural integrator with a two-body problem. Since the integrator is trained with three-body simulations, we want to see whether the NIH understands that a two-body problem can be seem as a three-body problem without a perturbing third body. To compare with existing works, we use the same initial condition as Greydanus et al. [2019]. We carry out the simulation for 1,000 time steps, and `WH-NIH` yields a relative energy error of $dE/E_0 \sim 10^{-9}$ by the end of the simulation. This result is about 7 orders of magnitude more accurate comparing to Greydanus et al. [2019].

3.2 The TRAPPIST-1 Exoplanet System

TRAPPIST-1 [Gillon et al., 2016] is an extrasolar planetary system with seven Earth-size planets. Because of its very special orbital architecture (compact super-Earth system with seven known planets orbiting a spectral type M8V star), it has since attracted considerable interests concerning its origin and potential for habitability. We test the `WH-NIH` integrator using the orbital parameters observed in TRAPPIST-1. The initial conditions of its orbital parameters are obtained from the exoplanets

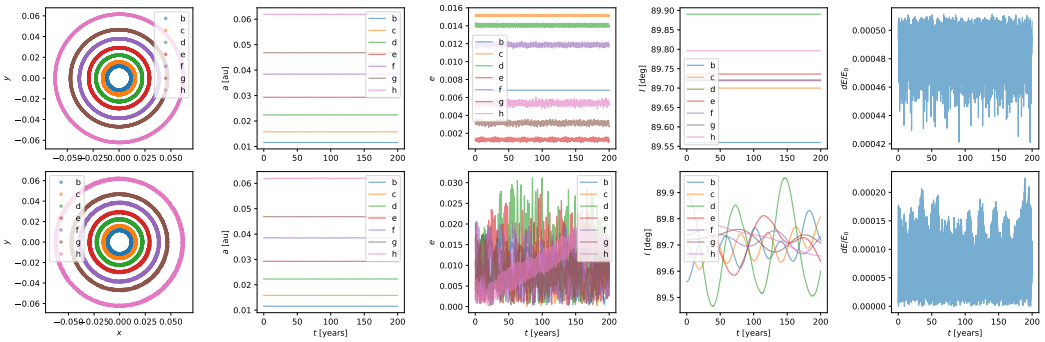

Figure 2: Evolution of the TRAPPIST-1 system for 200 years (equivalent to 2×10^5 time steps). First row: results from a Wisdom-Holman N-body integrator with neural interacting Hamiltonian; second row: ground truth results from a traditional N-body integrator. First column: orbit plots of Jupiter and Saturn; second column: orbital semi-major axes (a) as a function of time t; third column: orbital eccentricities e as a function of t; fourth column: orbital inclinations I (with respect to the heliocentric plane) as a function of t; last column: relative energy error (defined as the energy drift normalized to the initial energy) as a function of t. The legends "b", "c", ..., "h" represent the first, second, ..., and the seventh planet of the TRAPPIST-1 system, respectively.

database.[2] The innermost planet TRAPPIST-1b has an orbital period of 1.51 days, and its orbital semi-major axis of $a = 0.0011$ au is well below the minimum semi-major axes (0.4 au) in the training dataset, so the `WH-NIH` integrator has never been trained to handle such an extreme system. For this particular system, we will use an integration time step of $h = 0.365$ days.

As shown in Fig. 2, we integrate the system for 200 years, which corresponds to 2×10^5 integration time steps or 4.9×10^4 orbits of the innermost planet. Interestingly, even though the TRAPPIST-1 is well beyond the regime of the initial training data set for `WH-NIH`, the neural symplectic N-body integrator handles it nicely. The evolution of the Cartesian coordinates and semi-major axis are indistinguishable from the results of the numerical integration. Concerning the $t - e$ and $t - I$ plots, the `WH-NIH` integrator conserves the total angular momenta but fails to capture the rich dynamics of secular resonance between multiple planets. Nevertheless, we demonstrate that it is possible to learn the behavior of physical laws using a neural network and that the `WH-NIH` can apply the learned physical laws to entirely different systems.

4 Conclusions

We propose an efficient neural N-body solver based on the splitting the symplectic Hamiltonian map, as proposed by Wisdom and Holman [1991]: instead of approximating the full Hamiltonian $\mathcal{H}$, we approximate only a part of it and solve the other part analytically. In doing so, the error made by the DNN is constrained. We also develop a new activation function `SymmetricLog` that allows the neural integrator to work directly on the raw data without the need of pre-processing. We develop a custom loss function that takes into account the conservation of energy and angular momentum. We have demonstrated that our methods allows a DNN to solve the N-body problem for $\sim 10^5$ steps without diverging from the ground truth physics.

The proposed neural N-body solver is experimental, and there are some important limitations. If an N-body system has different interacting Hamiltonian, then the neural integrator will have to be retained. In addition, the neural integrator has been unable to understand the subtlety of secular angular momentum exchanges, and therefore it is not yet capable of modeling scenarios such as the eccentric Lidov-Kozai mechanism [see, e.g., Kozai, 1962, Naoz, 2016].

[2] `http://exoplanet.eu`

Acknowledgments and Disclosure of Funding

It is a pleasure to thank Sam Greydanus, Javier Roa, Adrian Hamers, Vikram A. Saletore, Capsar van Leeuwen, Valeriu Codareanu, and Veronica Saz Ulibarrena for insightful discussions. This research is partially sponsored by SURF Open Innovation Lab (SOIL) and Intel Corporation.

References

Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. Symplectic Recurrent Neural Networks. *arXiv e-prints*, art. arXiv:1909.13334, Sept. 2019.

D. Constantin Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta. Scalable Training of Artificial Neural Networks with Adaptive Sparse Connectivity inspired by Network Science. *arXiv e-prints*, art. arXiv:1707.04780, July 2017.

M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian Neural Networks. *arXiv e-prints*, art. arXiv:2003.04630, Mar. 2020.

Y. Desmond Zhong, B. Dey, and A. Chakraborty. Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control. *arXiv e-prints*, art. arXiv:1909.12077, Sept. 2019.

M. Erdmann, L. Geiger, J. Glombitza, and D. Schmidt. Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks. *arXiv e-prints*, art. arXiv:1802.03325, Feb. 2018.

M. Finzi, K. A. Wang, and A. G. Wilson. Simplifying Hamiltonian and Lagrangian Neural Networks via Explicit Constraints. *arXiv e-prints*, art. arXiv:2010.13581, Oct. 2020.

C. Fremling, X. J. Hall, M. W. Coughlin, A. S. Dahiwale, D. A. Duev, M. J. Graham, M. M. Kasliwal, E. C. Kool, A. A. Miller, J. D. Neill, D. A. Perley, M. Rigault, P. Rosnet, B. Rusholme, Y. Sharma, K. M. Shin, D. L. Shupe, J. Sollerman, R. S. Walters, and S. R. Kulkarni. SNIascore: Deep Learning Classification of Low-Resolution Supernova Spectra. *arXiv e-prints*, art. arXiv:2104.12980, Apr. 2021.

M. Gillon, E. Jehin, S. M. Lederer, L. Delrez, J. de Wit, A. Burdanov, V. Van Grootel, A. J. Burgasser, A. H. M. J. Triaud, C. Opitom, B.-O. Demory, D. K. Sahu, D. Bardalez Gagliuffi, P. Magain, and D. Queloz. Temperate Earth-sized planets transiting a nearby ultracool dwarf star. *Nature*, 533(7602):221–224, May 2016. doi: 10.1038/nature17448.

S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian Neural Networks. *arXiv e-prints*, art. arXiv:1906.01563, June 2019.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, art. arXiv:1412.6980, Dec. 2014.

Y. Kozai. Secular perturbations of asteroids with high inclination and eccentricity. *AJ*, 67:591–598, Nov. 1962. doi: 10.1086/108790.

S. Naoz. The Eccentric Kozai-Lidov Effect and Its Applications. *ARA&A*, 54:441–489, Sept. 2016. doi: 10.1146/annurev-astro-081915-023315.

J. Pieterse and D. Constantin Mocanu. Evolving and Understanding Sparse Deep Neural Networks using Cosine Similarity. *arXiv e-prints*, art. arXiv:1903.07138, Mar. 2019.

S. K. N. Portillo, J. K. Parejko, J. R. Vergara, and A. J. Connolly. Dimensionality Reduction of SDSS Spectra with Variational Autoencoders. *AJ*, 160(1):45, July 2020. doi: 10.3847/1538-3881/ab9644.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv e-prints*, art. arXiv:1711.10561, Nov. 2017a.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv e-prints*, art. arXiv:1711.10566, Nov. 2017b.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

I. Reis, D. Poznanski, D. Baron, G. Zasowski, and S. Shahaf. Detecting outliers and learning complex structures with large spectroscopic surveys - a case study with APOGEE stars. *MNRAS*, 476(2):2117–2136, May 2018. doi: 10.1093/mnras/sty348.

A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia. Hamiltonian Graph Networks with ODE Integrators. *arXiv e-prints*, art. arXiv:1909.12790, Sept. 2019.

P. Toth, D. Jimenez Rezende, A. Jaegle, S. Racanière, A. Botev, and I. Higgins. Hamiltonian Generative Networks. *arXiv e-prints*, art. arXiv:1909.13789, Sept. 2019.

J. Wisdom and M. Holman. Symplectic maps for the N-body problem. *AJ*, 102:1528–1538, Oct. 1991. doi: 10.1086/115978.

J. Zamudio-Fernandez, A. Okan, F. Villaescusa-Navarro, S. Bilaloglu, A. Derin Cengiz, S. He, L. Perreault Levasseur, and S. Ho. HIGAN: Cosmic Neutral Hydrogen with Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1904.12846, Apr. 2019.

A Appendix

A.1 Initial Conditions of the Simulations

To generate training data for the NIH, we carry out 50 three-body simulations with random initial conditions. For simplicity, we define the central body as a $1 M_\odot$[3] star. The masses of the two orbiters vary from 1 $M_\odot$ (two equal-mass stars orbit around each other, also known as a binary stellar system), $10^{-3} M_\odot$ (a Jupiter-mass planet orbiting around the Sun), $10^{-6} M_\odot$ (an Earth-mass planet orbiting around the Sun) to $10^{-8} M_\odot$ (a moon-mass object orbiting around the Sun). The orbital semi-major axes ranges from 0.4 au[4] (comparable to Mercury's orbit) to 100 au (comparable to 3 times the size of Neptune's orbit). For convenience, we set $G = 4\pi^2$ as the value of the universal gravitational constant,[5] unless otherwise noted.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the discussion in each subsection in Section 3.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2 for the assumptions of the underlying theoretical framework.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 2 for the derivation of the equations.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The source code and data analysis scripts will be hosted on a GitHub repository.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] No, the result presentation is example-based, but the neural network predictions are compared directly with the ground truth. See Fig 2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 2, where it states that the training can be done on a single GPU or CPU within 1 hour.

[3] $1 M_\odot \approx 1.9985 \times 10^{30}$ kg is the mass of the Sun in our Solar System.
[4] 1 au $\approx 1.496 \times 10^8$ km is the time-averaged distance from the Sun to the Earth.
[5] When $G = 4\pi^2$ the corresponding mass unit is $M_\odot$, length unit is au, and time unit is years.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]