"Pseudo-Hamiltonian Neural Networks for Partial Differential Equations" a summary of background, results, and applications

Control and Optimization Seminar,

Technische Universität Berlin, Summer 2024

Dimitri Alexander Cacouris

July 25, 2024

1 Introduction

This is a summary of Eidnes & Lye's 2024 article, prepared along with an accompanying presentation in the context of the seminar "Control and Optimization" at the TU Berlin during the Summer 2024 semester.
A recent development in the domain of physics-informed machine learning (and therefore related disciplines), Hamiltonian neural networks were proposed in 2019 by Greydanus et al. during a Google AI Residency. This 2024 article is an extension and direct follow-up of a 2023 paper by Eidnes and collaborators at SINTEF, and so given the foundations laid there, along with the (recent) historical context in which this approach was developed, a (relatively) lengthy amount of attention will be given to both the 2019 and 2023 antecedents, before the extensions given in the 2024 paper are discussed.

2 Understanding the Title, i.e. Background

Fortunately, the title of this article belongs to that genre which is both clear and illustrative not only of its contents, but also of the necessary background we require to understand them. The first question even a reader familiar with the field is likely to ask themselves is: what is meant by a "pseudo-Hamiltonian"?

2.1 Hamilton for Hamiltonians

As it will certainly be overly didactic to introduce the concept as if for novices, we will merely revisit here the key concepts and expressions of the Hamiltonian, which will be necessary for making the approach below clear.
A modified form of the Lagrangian (via Legendre transformation) of classical mechanics, the Hamiltonian likewise captures the dynamics of the system, classically in terms of generalised coordinates q and associated momenta $p = \dot{q}$. Since for physical systems energy and momentum must be conserved, the Hamiltonian defines 'physically plausible' paths and therefore the system evolution. The key formulation to note here is actually the simplest and most most classical, namely:

$$\frac{\partial \mathcal{L}}{\partial q} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} = 0 \quad \Rightarrow \quad \frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial \mathcal{H}}{\partial p}, \quad \frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial \mathcal{H}}{\partial q} \tag{1}$$

2.2 Port-Hamiltonian & Pseudo-Hamiltonian Formalisms

Developed out of port-based modeling, which "recogni[ses] energy as the *lingua franca* between physical systems"[1], the port-Hamiltonian formalism allows modeling of a physical system as a series of interconnected modules, between which energy must be conserved, including modules representing external forces and dissipative interactions, meaning that isolated (parts of) systems can be modeled with energy in- and outflow while retaining the strict conservation of the Hamiltonian formalism. For this purpose, the geometric Dirac structure is often employed, but we will follow the equivalent algebraic formulations of the papers under examination. Likewise, the gradual construction of the proposed 'pseudo-Hamiltonian' formulation as done by the authors, modifying the classical form to arrive at the generalisation, is most illustrative.
Let us therefore begin by observing that using $x = \begin{pmatrix} q & p \end{pmatrix}^T \in \mathbb{R}^{2n \times 2n}$, the classical set of equations (1) can be rewritten as

$$\dot{x} = J\nabla\mathcal{H}, \quad \text{with } J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix},$$

but we can substitute x with elements of any desired state space that will capture our system dynamics. This canonical form can be extended to a non-canonical form by substituting any other skew-symmetric—and possibly state-dependent—matrix $S(x) \in \mathbb{R}^{2n \times 2n}$ for J to get

$$\dot{x} = S(x)\nabla\mathcal{H},$$

and this formulation will exist (albeit with S not unique for any $n > 1$) for any $\mathcal{H}$ that fulfills $\frac{d\mathcal{H}}{dt} = \nabla\mathcal{H}(x)^T g(x) = 0$.
The authors then extend this by including a second state-dependent matrix R and function $f(x,t)$ to arrive at what they term their 'pseudo-Hamiltonian' formulation:

$$\dot{x} = (S(x) - R(x))\nabla\mathcal{H} + f(x,t), \tag{2}$$

with $R(x) \in \mathbb{R}^{2n\times 2n}$ positive semi-definite. This resembles the port-Hamiltonian formalism, with S representing energy-storing (i.e. conserved) elements, R resistive (i.e. dissipative) elements, and f forces wholly external to the system under consideration, from which energy could enter or exit it. However, unlike in Van der Schaft & Jeltsema**??**, the authors assume no particular structure on f, notably not restricting systems to be passivity-preserving. They also note that it is quite close to the GENERIC formalism proposed by Grmela & Öttinger, and in fact the partial differential version will resemble it even further in its separation of $\mathcal{H}$ into two possibly different functions. The approach here is intended to be a broad formulation to implement in their neural network design, and from which narrower restrictions will be made depending on the system in question. [2]

3 Neural Networks

We now turn (briefly) to the second term in the title, though as whole tomes can be devoted to the topic, we will likewise only consider a few key features relevant to the implementations in the three papers. A neural network is essentially a large composition of simple functions, each representing an artifical (and almost always emulated) 'neuron', usually summed or convoluted in several 'hidden' layers, each of which takes inputs from the preceding layer, and ultimately the initial input, and outputing them in the last layer. Despite the activation functions for each neuron being exceedingly simple—usually the Heaviside step function, the rectifier, hyperbolic tangent, logistic sigmoid, or similar—universal approximation theorems guarantee the arbitrarily close approximation of arbitrary continuous functions by a (sufficiently large) neural network of non-polynomial activation functions. The representation of inputs and operations as tensors and products therof allows for efficient computation of outputs and training the network through backpropigation. Learning is usually accomplished by measuring the output against a loss (i.e. error) function, and using the chain rule to adjust the weights of the network in the direction of the gradient; as the loss function is minimised, its design essentially defines what the network learns. Additionally, Eignes & Lye note that the convolutions of convulutional neural networks resemble finite difference approximations:

$$\text{Compare} \quad (u * w)(x_i) \coloneqq \sum_{j=-r}^{s} w_j u(x_{i-j}) \quad \text{with} \quad \frac{d^n u(x_i)}{dx^n} \approx \sum_{j=-r}^{s} a_j u(x_i - j)$$

These properties are important to ensure that, having defined a (partial) differential equation we wish to model, a neural network is actually capable of approaching it, and doing so in a remotely efficient manner.

3.1 Hamiltonian Neural Networks

In their 2019 paper "Hamiltonian Neural Networks", Sam Greydanus, Jason Yosinski, and Misko Dzamba made use of these properties to propose the idea of "Instead of crafting the Hamiltonian by hand,[. . .] parameterizing it with a neural network and then learning it directly from data."[3] While for simple systems with well-studied dynamics, the Hamiltonian might be well-known, for systems with uncertain dynamics or measured through noisy data, deriving the system dynamics can be quite difficult. They therefore designed the following loss function, in which $\mathcal{H}_\theta$ is a neural network parametrised by θ:

$$\mathcal{L}_{HNN} = \left\| \frac{\partial \mathcal{H}_\theta}{\partial p} - \frac{\partial q}{\partial t} \right\|^2 + \left\| \frac{\partial \mathcal{H}_\theta}{\partial q} - \frac{\partial p}{\partial t} \right\|^2$$

Comparing with (1) above, we can see that the Hamiltonian is simply and directly incorporated therein, so that a minimal $\mathcal{H}_{\hat{\theta}} \coloneqq \operatorname*{argmin}_{\mathcal{H}_\theta} \mathcal{L}_{HNN}$ represents perfectly conserved energy and an idealised system path. The authors tested this approach on a couple simple systems, both simulated and real, before trying the approach on two- and three-body systems, as well as extending the framework to draw data directly from images and add in external energy impulses. In comparing the results with a baseline neural network, they found that the baseline model seemed to 'lose energy', diverging strongly from both idealised predictions and reality, whereas the Hamiltonian neural network (HNN) almost perfectly conserved the energy and followed idealised predictions, though it not surprisingly did not account for the real-world energy loss in the real example. It also outperformed the baseline model in the n-body systems, but did not capture the three-body-problem as well as the two-body problem (perhaps also not surprisingly). This paper nevertheless represented more of a proof-of-concept, both that the idea was sound, and that it could outperform conventional neural networks. The authors also published their framework as a Python package, hamiltonian-nns, available on GitHub.

4 Pseudo-Hamiltonian Neural Networks with State-Dependent External Forces

In this paper from the previous year, Sølve Eidnes, Alexander J. Stasik, Camilla Sterud, Eivind Bøhn, and Signe Riemer-Sørensen adapt HNNs to the general formulation of (2), and couple the more generalised approach to the following innovation: rather than modeling the system dynamics all together, they build separate neural networks $\hat{\mathcal{H}}_\theta$ and $\hat{f}_\theta$ to model $\mathcal{H}$ and f, respectively. A joint model they call a pseudo-Hamiltonian Neural Network (PHNN)

$$\hat{g}_\theta \coloneqq (S - \hat{R}_\theta)\nabla\hat{\mathcal{H}}_\theta(x) + \hat{f}_\theta(x, t)$$

is then trained using the discretised loss function

$$\mathcal{L} = \left\| \frac{x^{n+1} - x^n}{\Delta t} - \hat{g}_\theta \left(\frac{x^n + x^{n+1}}{2}, \frac{t^n + t^{n+1}}{2} \right) \right\|_2^2 + \frac{\lambda}{N} \left\| \hat{f}_\theta \left(\frac{x^n + x^{n+1}}{2}, \frac{t^n + t^{n+1}}{2} \right) \right\|_1,$$

an implicit midpoint method with λ a regularisation parameter[1]. Note that for the examples actually considered, they assume the state-independence of both S and R, but include parameters of R to be learned, hence "S" but "$\hat{R}_\theta$".
The idea thereby is to learn the internal dynamics and internal energy separately from the external (also state-dependent) dynamics and external energy, essentially using neural networks to mirror a port-Hamiltonian model, albeit with a slightly generalised formulation.[2] The authors therefore orient their approach on addressing systems "when the external forces may be state-dependent and the change in energy stemming from damping and external forces cannot immediately be separated." [2] They also caution that their formulation renders solutions to the system, and importantly the separation of internal and external dynamics, non-unique, meaning that assumptions or prior information about the system may have to be incorporated to isolate the desired results. Given that it is "difficult to provide guarantees on the PHNN model in its most general form" [2], they rather suggest an iterative approach to using it to learn system dynamics, first beginning with a more general formulation, and using knowledge gained from prior iterations to encode more restrictions into both networks for improved results. As with Greydanus et al., a Python package with this implementation, phlearn, has been made available on GitHub.

5 Eidnes & Lye

5.1 Approach

The 2024 paper ([4], note all citations in this section derive from here) treads much of the same ground as the 2023 paper from the SINTEF team, its chief contribution being to extend the ordinary differential formulation to a partial differential one, and thereby to complicate the neural network(s) used, going from two to six possible independently[3] trainable models. They first of all open up consideration of separate dynamics affecting the energy-preserving and energy-dissipating elements and allow considering infinite-dimensional systems, replacing $\mathcal{H}$ alone with $\mathcal{H}$ and $\mathcal{V}$, in addition to considering the partial spatial derivatives $u^a := \left\{ \frac{\partial^\alpha u}{\partial x_1^{\alpha_1} \cdots x_d^{\alpha_d}} : |\alpha| \leq p \right\}, \alpha \in (\mathbb{Z}^{\geq})^d$. The ODE (2) is thus broadened to the following class of PDEs:

$$u_t = S(u^a, x) \frac{\delta \mathcal{H}}{\delta u}[u] - R(u^a, x) \frac{\delta \mathcal{V}}{\delta u}[u] + f(u^a, x, t).$$

As before, S is skew-symmetric and R is positive semi-definite. Note that now both $\mathcal{H}[\cdot]$ and $\mathcal{V}[\cdot]$ are spacial integrals of the form $\mathcal{H}[u] = \int_\Omega H(x, u, u_x)$ for a function H, the derivative of which for integrals only dependent on first derivatives is given by $\frac{\delta \mathcal{H}}{\delta u}[u] = \frac{\partial \mathcal{H}}{\partial u} - \frac{d}{dx} \frac{\partial \mathcal{H}}{\partial u_x}$, giving us a familiar formulation in its relation to (1). Again, this is an extremely general approach not designed to be useful in itself, but rather to allow the design of a general neural network framework that, with the appropriate restrictions, could be used to learn "Hamiltonian, port-Hamiltonian, dissipative and metriplectic PDEs, with and without external forces".[4]

5.2 Restrictions

Indeed, to design their neural network as desired, the authors make the following (much heavier) restrictions:

- S and R are both linear, and independent of both x and u
- R is symmetric
- f is independent of derivatives of u
- The symmetric, positive semi-definite operator A commutes with both R and S, so that they can rewrite $R := AR$, $S := AS$, $f := Af$, and the PDE as a whole as:
$$Au_t = S \frac{\delta \mathcal{H}}{\delta u}[u] - R \frac{\delta \mathcal{V}}{\delta u}[u] + f(u, x, t)$$

Should R be a linear combination of the identity and differential operators, the separation of it with the external force f cannot be determined beyond scaling of a constant; not a problem for making sense of the model, but while training the neural network, this constant can "leak" between models, causing overfitting. To avoid this, they assume that R will be

[1]they also propose a second, better-performing yet more computationally expensive method for use with sparser data "comparable to the classic Runge-Kutta method" [2]

[2]They are also following in the footsteps of work by S.A. Desai et al. in 2021 (which won't be addressed here), in which a similar separation was done, but for systems with only time-dependent external forces.

[3]that is, independently input into the total model and stored, not with independent loss functions

[4]As mentioned, this class is even more similar to the GENERIC formalism, differing by the inclusion of f, requiring R to be positive instead of negative semi-definite, and not requiring degeneracy conditions. The authors also note the similarity (again other than f) to metriplectic PDEs, and that for $\mathcal{V} = 0$, $f = 0$ the class is that of integral-preserving PDEs, which includes all non-canonical Hamiltonian PDEs; if S also satisfies the Jacobi identity, then $\mathcal{H}$ directly represents the Hamiltonian. Alternatively, for $\mathcal{H} = 0$, $f = 0$, $\mathcal{V} \geq 0$, the PDE will dissipate $\mathcal{V}$, making it a Lyapunov function.

zero for the zero solution, and adjust both $\hat{R}_\theta$ and $\hat{f}_\theta$ after training to ensure this while maintaining an equivalent model. They caution, however, that this correction will not be effective, should the zero solution be "far outside" the training data. Additionally, for the numerical experiments done in the paper, A and R are never more than "linear combinations of the identity and the spacial second derivative" (necessitating the above considerations), nor is S more than the first spacial derivative. Moreover, for some experiments, these matrices are assumed to be known explicitly, with a marked effect on the results (see below).

5.3 Neural Network Model

The strong restrictions, particularly the independence of x and u allow A, S, and R to "be viewed as discrete convolution operators" and to form analogous neural networks $\hat{A}_\theta^{[k_1]}$, $\hat{S}_\theta^{[k_2]}$, $\hat{R}_\theta^{[k_3]}$, in addition to $\hat{\mathcal{H}}_\theta$, $\hat{\mathcal{V}}_\theta$, $\hat{f}_\theta$ for six possible trainable neural networks, which are fed into the total model:

$$\hat{g}_\theta(u, x, t) = (\hat{A}_\theta^{[k_1]})^{-1}\left(\hat{S}_\theta^{[k_2]}\nabla\hat{\mathcal{H}}_\theta(u) - \hat{R}_\theta^{[k_3]}\nabla\hat{\mathcal{V}}_\theta(u) + k_4\hat{f}_\theta(u, x, t)\right)$$

The terms k_i are kernel sizes, with the exception of $k_4 \in \{0, 1\}$, used to control the presence of external forces, and comprise input variables to the setup of the neural network. Analogously to the previous paper, an implicit midpoint integrator defines the loss function

$$\mathcal{L}_{g_\theta}(\{(u^{j_n}, u^{j_{n+1}}, t^{j_n})\}_{n=1}^N) = \frac{1}{N}\sum_{n=1}^{N}\left|\frac{u^{j_{n+1}} - u^{j_n}}{\Delta t} - \hat{g}_\theta\left(\frac{u^{j_n} - u^{j_{n+1}}}{2}, x, \frac{t^{j_n} - t^{j_{n+1}}}{2}\right)\right|^2$$

but allowances have been made to use any mono-implicit integrator, including the second one used in the previous paper.

5.4 Spacial discretisation

Given the new possibility of infinite-dimensional systems, discretisation has to be employed, and so the authors approximate the L_2 inner product with a weighted discrete inner product

$$\langle u, v\rangle = \int_\Omega u(x)v(x)dx \approx \sum_{i=0}^{M}\kappa_i u(x_i)v(x_i) = u^T diag(\kappa)v =: \langle u, v\rangle_\kappa,$$

where κ_i are non-zero quadrature weights for grid points $\mathbf{x} = [x_0 \cdots x_M]^T$. Using $\mathbf{p}$ for both of these discretisation parameters and assuming the existence of a consistent approximation $\mathcal{H}_\mathbf{p}(\mathbf{u})$ to $\mathcal{H}[u]$ (a much less egregious assumption with the existence of universal approximation theorems) with $\mathbf{u}$ evaluated at $\mathbf{x}$, the discretised variational derivative can be represented as $\frac{\delta\mathcal{H}_\mathbf{p}}{\delta\mathbf{u}}(\mathbf{u}) = \text{diag}(\kappa)^{-1}\nabla_\mathbf{u}\mathcal{H}_\mathbf{p}(\mathbf{u})$.

5.5 Performance

The authors have untertaken several numerical experiments, including with the KdV, Perona-Malik, and Cahn-Hillard equations, to test the performance of their approach in comparison with baseline (conventional) nerual networks as well as models specifically designed for identifing PDEs: DGNet and PDE-FIND. They also split their approach into different versions: a "general", i.e. fully 'naïve', one that tries to identify A, S, and R and leaves $\hat{f}_\theta$ dependent on (u, x, t) , a "lean" one without S and smaller kernels, one without R, and an 'informed' one, in which they explicitly set all three matrices ahead of time and set $\hat{f}_\theta$ to have the dependencies of the true f, a hefty 'head start' in model identification.
Qualitatively, the "general" model does always seem to do at least as well as the baseline, sometimes barely, but frequently categorically better, approaching the true solution where the baseline produces insensible or diffuse outputs. Not surprisingly, the "informed" model usually performs much better than the others[5], with a mean squared error often an order of magnitude better than the runner-up non-PHNN. As with many neural networks, the PHNNs are quite sensitive to their initialisation, though they find both greater stability compared to the baseline and a quicker convergence to the ground truth. Given the frequent (though not always) markedly better performance of these "informed" models, the authors as in the previous article envision an iterative technique with their approach, feeding insights gleaned from the "general" model into increased restrictions for subsequent iterations, until the models begin to converge.

6 Results, Applications, & Critique

So how can we summarise the results and insights of both this publication and the HNN concept as a whole? Positive features include:
- HNNs seem to learn physical systems better than conventional neural networks.

[5]In fact, where a "purely Hamiltonian" system is examined, the "informed" model is then equivalent to a(n ODE) HNN.

- The pseudo-Hamiltonian model extends this benefit to non-canonical Hamiltonian systems, and now to those with a PDE form.
- The separation of internal dynamics, dissipation, and external forces not only allows incorporation of preexisting knowledge and (even post-hoc) alteration of parts of the system without disturbing correctly-learned others, but may allow for "meaningful physical interpretations" [2] to be made from these parts and their interactions.
- Plus, the effects of "unknown external forces" [2] could be detected, drawing attention to a needed object of investigation.

Some (current) limitations include:

- PHNNs do not always outperform other neural networks, even conventional ones.[6]
- Though the PDE pseudo-Hamiltonian model is indeed usefully general, the strong restrictions required to create the PHNN seem to mitigate this initial generality.
- Likewise, though for some cases the neural network works 'out of the box', a degree of prior knowledge will often be required.
- Like many NNs, PHNNs are highly sensitive to their initialised parameters.

In addition to these general limitations, the tests conducted in all three articles were chiefly toy examples of well-known systems, and though the PDE extension also allows infinite-dimensional systems, only one-dimensional systems were considered; a natural first step for a proof-of-concept, but testing on more complicated examples would be a wise second one.

6.1 Applications

In addition to the backward problem of modeling physical systems given (noisy) data, the authors also offer the utility of solving PDEs like the heat or Perona-Malik equations in image denoising, and the Cahn-Hillard or Allen-Cahn equations in phase separation and pattern formation. The initial citations made of this paper at the time of writing (July 2024) suggest that the first application will continue to be the chief one, though.

6.2 Next Steps

In addition to trying more complicated and higher-dimensional examples, trying the package(s) on more real-world (and noisier) data sets would be useful for refining this approach, as most trials have been done with simulated data and noise. The authors themselves claim they plan to improve the models and associated Python package with regard to stability and consistency in the discretisation methods, and suggest that their pseudo-Hamiltonian formulation might find applications in machine learning models other than neural networks as well. The overall scheme (both of HNNs and PHNNs) of beginning with priors of the most general formulation and refining these as aspects of the system becomes clearer likewise recommend themselves for use when modeling things other than physical systems, or outside the domain of ML all together, as long as the general priors can be formulated in a way which is strictly known to be true (as energy conservation is in the Hamiltonian). Combining this approach with others used for identifying and classifying physical systems, such as the equally-young approach of recurrence microstates, might also be a fruitful avenue.

References

[1] A. J. v. d. Schaft and D. Jeltsema, *Port-Hamiltonian systems theory: an introductory overview*, ser. Foundations and trends in systems and control. Boston Delft: Now, 2014, no. 1, 2/3 (2014).

[2] S. Eidnes, A. J. Stasik, C. Sterud, E. Bøhn, and S. Riemer-Sørensen, "Pseudo-Hamiltonian Neural Networks with State-Dependent External Forces," *Physica D: Nonlinear Phenomena*, vol. 446, p. 133673, Apr. 2023, arXiv:2206.02660 [cs, math]. [Online]. Available: http://arxiv.org/abs/2206.02660

[3] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian Neural Networks," Sep. 2019, arXiv:1906.01563 [cs]. [Online]. Available: http://arxiv.org/abs/1906.01563

[4] S. Eidnes and K. O. Lye, "Pseudo-Hamiltonian neural networks for learning partial differential equations," Jan. 2024, arXiv:2304.14374 [cs, math]. [Online]. Available: http://arxiv.org/abs/2304.14374

[6] Looking at the apparent dissipation included in conventional NNs in comparison to HNNs, one might be tempted to wonder if these might even be worth employing in certain situations, as a kind of 'fuzzy logic'. While the uncontrolled nature of this approximation argues for the genuine advantage of (P)HNNs, the approach may have to improve further to compete against highly generalised NNs that may often be seen as 'good enough'.