Leveraging Hamilton-Jacobi PDEs with time-dependent Hamiltonians for continual scientific machine learning

Paula Chen* PAULA.X.CHEN.CIV@US.NAVY.MIL
Naval Air Warfare Center Weapons Division, China Lake, CA 93555, USA

Tingwei Meng* TINGWEI@MATH.UCLA.EDU
Department of Mathematics, UCLA, Los Angeles, CA 90025, USA

Zongren Zou* ZONGREN_ZOU@BROWN.EDU
Division of Applied Mathematics, Brown University, Providence, RI 02912, USA

Jérôme Darbon JEROME_DARBON@BROWN.EDU
Division of Applied Mathematics, Brown University, Providence, RI 02912, USA

George Em Karniadakis GEORGE_KARNIADAKIS@BROWN.EDU
Division of Applied Mathematics and School of Engineering, Brown University, Providence, RI 02912, USA
Pacific Northwest National Laboratory, Richland, WA 99354, USA

Abstract

We address two major challenges in scientific machine learning (SciML): interpretability and computational efficiency. We increase the interpretability of certain learning processes by establishing a new theoretical connection between optimization problems arising from SciML and a generalized Hopf formula, which represents the viscosity solution to a Hamilton-Jacobi partial differential equation (HJ PDE) with time-dependent Hamiltonian. Namely, we show that when we solve certain regularized learning problems with integral-type losses, we actually solve an optimal control problem and its associated HJ PDE with time-dependent Hamiltonian. This connection allows us to reinterpret incremental updates to learned models as the evolution of an associated HJ PDE and optimal control problem in time, where all of the previous information is intrinsically encoded in the solution to the HJ PDE. As a result, existing HJ PDE solvers and optimal control algorithms can be reused to design new efficient training approaches for SciML that naturally coincide with the continual learning framework, while avoiding catastrophic forgetting. As a first exploration of this connection, we consider the special case of linear regression and leverage our connection to develop a new Riccati-based methodology for solving these learning problems that is amenable to continual learning applications. We also provide some corresponding numerical examples that demonstrate the potential computational and memory advantages our Riccati-based approach can provide.

Keywords: Hamilton-Jacobi PDEs; generalized Hopf formula; continual learning; optimal control

1. Introduction

Scientific machine learning (SciML) encompasses a wide range of powerful, data-driven techniques renowned for their ability to solve complicated problems that more traditional numerical methods cannot. Despite these successes, developing the theoretical foundations of SciML is still an active area of research (e.g., see Carvalho et al. (2019)). In this work, we increase the interpretability of certain learning processes by establishing a new theoretical connection between certain optimization problems arising from SciML and a generalized Hopf formula, which represents the viscosity solution to a Hamilton-Jacobi partial differential equation (HJ PDE) with time-dependent Hamiltonian (Section 2). HJ PDEs have been shown to have deep connections with many scientific disciplines, including but not limited to optimal control (Bardi and Capuzzo-Dolcetta (1997)) and differential

* These authors contributed equally to this work.

games (Evans and Souganidis (1984)). Here, we leverage our new theoretical connection as well as the established connection between HJ PDEs and optimal control to show that when we solve certain regularized learning problems, we actually solve an optimal control problem and its associated HJ PDE with time-dependent Hamiltonian. In doing so, we can reuse existing efficient HJ PDE solvers and optimal control algorithms to develop new training approaches for SciML. We also build upon our previous work in Chen et al. (2024a), which, to our knowledge, is the first to establish a connection of this kind between SciML and HJ PDEs. In Chen et al. (2024a), we instead connect regularized learning problems with the multi-time Hopf formula, which represents the solution to certain multi-time HJ PDEs (Rochet (1985); Lions and Rochet (1986)). By discretizing the generalized Hopf formula (e.g., by discretizing the integral in the middle row of Figure 1), we can regard the work here as a generalization of Chen et al. (2024a) to the infinite-time case.

By instead considering HJ PDEs with time-dependent Hamiltonians, we develop a new theoretical framework that is more closely aligned with continual learning (Parisi et al. (2019); Van de Ven and Tolias (2018)), which in turn yields potential computational and memory advantages, especially in big data regimes. Under the continual learning framework, data is accessed in a stream and learned models are updated incrementally as new data becomes available. In many continual learning scenarios, data is also assumed to become inaccessible after it is incorporated into the learned model, which often leads to catastrophic forgetting (Kirkpatrick et al. (2017); Parisi et al. (2019)) or, in other words, the abrupt degradation in the performance of learned models on previous tasks upon training on new tasks. However, this lack of dependence on historical data often also facilitates the development of more computationally efficient learning algorithms by requiring that only a small amount of data can be processed and stored at a time. Using our theoretical connection, we reinterpret incrementally updating learned models as evolving an associated HJ PDE and optimal control problem in time, where all of the information from previous data points are inherently encoded in the solution to the HJ PDE. As such, our new interpretation has the potential to yield new efficient continual learning approaches that naturally avoid catastrophic forgetting.

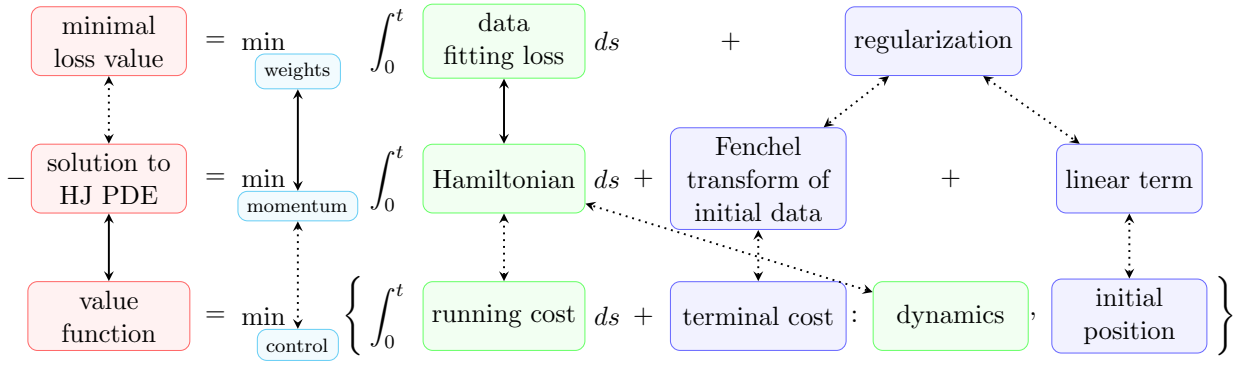


Figure 1: (See Section 2) Illustration of a connection between a regularized learning problem with integral-type loss (**top**), the generalized Hopf formula for HJ PDEs with time-dependent Hamiltonians (**middle**), and the corresponding optimal control problem (**bottom**). The colors indicate the associated quantities between each problem. For example, the optimal weights in the learning problem are equivalent to the momentum in the HJ PDE, which is related to the control in the optimal control problem (cyan). This color scheme is reused in the subsequent illustrations of our connection. The solid-line arrows denote direct equivalences. The dotted arrows represent additional mathematical relations.

As a first exploration of this connection, we consider the special case of regularized linear regression problems with integral-type data fitting losses, which we show are connected to linear quadratic regulator (LQR) problems with time-dependent costs and dynamics (Anderson and Moore (2007); Tedrake (2023)) (Section 3). We then leverage this connection to develop a new Riccati-based methodology that coincides with the continual learning framework, while inherently avoiding catastrophic forgetting. Finally, we demonstrate how this methodology can be applied to some SciML applications of interest (Section 4) and discuss some possible future directions (Section 5).

2. Connection between the generalized Hopf formula and learning problems

In this section, we begin by introducing the generalized Hopf formula and reviewing the well-established connections between HJ PDEs and optimal control. Then, we introduce the learning problems of interest and formulate our new theoretical connection.

2.1. Generalized Hopf formula for HJ PDEs with time-dependent Hamiltonians

Consider the following HJ PDE with time-dependent Hamiltonian:

$$\begin{cases} \dfrac{\partial S(\mathbf{x},t)}{\partial t} + H(t, \nabla_{\mathbf{x}} S(\mathbf{x},t)) = 0 & \mathbf{x} \in \mathbb{R}^n, t > 0, \\ S(\mathbf{x},0) = J(\mathbf{x}) & \mathbf{x} \in \mathbb{R}^n, \end{cases} \tag{1}$$

where $H : (0,\infty) \times \mathbb{R}^n \to \mathbb{R}$ is the Hamiltonian and $J : \mathbb{R}^n \to \mathbb{R}$ is the initial condition. Assume that H, J are continuous, J is convex, and $H(t,\mathbf{p})$ is convex in $\mathbf{p}$. Then, using the same proof as that for (Lions and Rochet, 1986, Proposition 1), it can be shown that the viscosity solution to this HJ PDE (1) can be represented by the following generalized Hopf formula:

$$S(\mathbf{x},t) = \sup_{\mathbf{p}\in\mathbb{R}^n} \left\{ \langle \mathbf{x},\mathbf{p}\rangle - \int_0^t H(s,\mathbf{p})ds - J^*(\mathbf{p}) \right\} = - \inf_{\mathbf{p}\in\mathbb{R}^n} \left\{ \int_0^t H(s,\mathbf{p})ds + J^*(\mathbf{p}) - \langle \mathbf{x},\mathbf{p}\rangle \right\}, \tag{2}$$

where f^* denotes the Fenchel-Legendre transform of the function f; i.e., $f^*(\mathbf{p}) = \sup_{\mathbf{x}\in\mathbb{R}^n}\{\langle \mathbf{x},\mathbf{p}\rangle - f(\mathbf{x})\}$. Note that, as stated in a remark in Lions and Rochet (1986), this Hopf representation formula does not hold in general. However, under our convexity assumptions, the semigroup commutation still holds and hence, we can follow the same proof as that for (Lions and Rochet, 1986, Proposition 1). More details are provided in the arXiv version of this paper (Chen et al. (2024b)). Also note that if the integral in (2) is discretized in time, then we recover a multi-time Hopf formula (e.g., see Rochet (1985); Lions and Rochet (1986); Chen et al. (2024a); Darbon and Meng (2020)), and hence, (1) can also be considered as a multi-time HJ PDE with infinitely many times.

It is well-known that the value function of the following optimal control problem also satisfies (1):

$$S(\mathbf{x},t) = \min_{\mathbf{u}(\cdot)} \left\{ \int_0^t L(s,\mathbf{u}(s))ds + J(\boldsymbol{x}(t)) : \dot{\boldsymbol{x}}(s) = f(s,\mathbf{u}(s)) \forall s \in (0,t], \boldsymbol{x}(0) = \mathbf{x} \right\}, \tag{3}$$

where $\mathbf{x}$ is the initial position, t is the time horizon, $\mathbf{u} : [0,\infty) \to \mathbb{R}^m$ is the control, $\boldsymbol{x} : [0,\infty) \to \mathbb{R}^n$ is the trajectory, the running cost L and the source term f of the dynamics are related to the Hamiltonian H by $H(s,\mathbf{p}) = \sup_{\mathbf{u}\in\mathbb{R}^m}\{\langle -f(s,\mathbf{u}),\mathbf{p}\rangle - L(s,\mathbf{u})\}$, and J is now the terminal cost.

2.2. Connection to learning problems

In this section, we connect the Hopf formula (2) to regularized learning problems with integral-type losses (e.g. see Sirignano and Spiliopoulos (2018)). Consider a learning problem with data

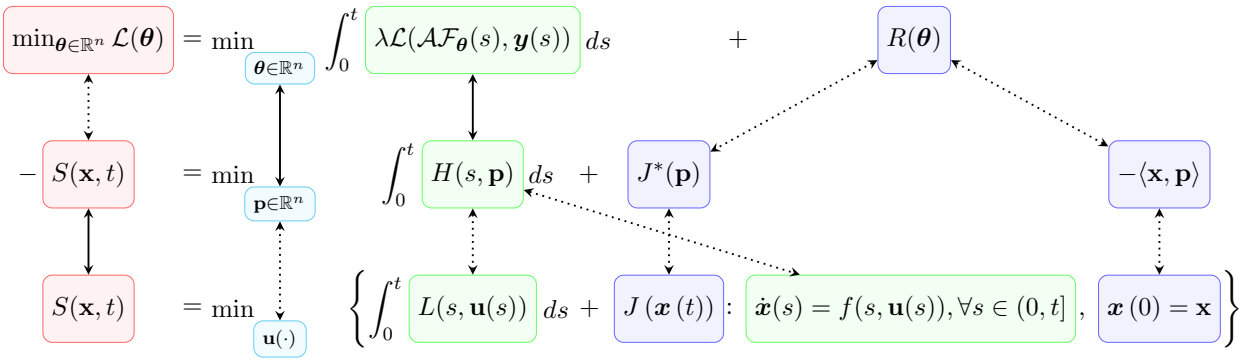


Figure 2: (See Section 2) Mathematical formulation describing the connection between a regularized learning problem with integral-type loss (**top**), the generalized Hopf formula for HJ PDEs with time-dependent Hamiltonians (**middle**), and the corresponding optimal control problem (**bottom**). The content of this illustration matches that of Figure 1 by replacing each term in Figure 1 with its corresponding mathematical expression. The colors indicate the associated quantities between each problem. The solid-line arrows denote direct equivalences. The dotted arrows represent additional mathematical relations.

$(s, \boldsymbol{y}(s)) \in [0,t] \times \mathbb{R}^m$. The goal is to learn a function $F(\cdot;\boldsymbol{\theta})$ with inputs in $[0,t]$ and unknown parameter $\boldsymbol{\theta} \in \mathbb{R}^n$. We learn $F(\cdot;\boldsymbol{\theta})$ as follows. First, we apply an operator $\mathcal{A}$ on the function $F(\cdot;\boldsymbol{\theta})$, where we denote $\mathcal{A}\mathcal{F}_{\boldsymbol{\theta}}(s) := \mathcal{A}[F(\cdot;\boldsymbol{\theta})](s)$. For example, $\mathcal{A}$ could be the identity operator (as in regression problems; e.g., see Weisberg (2005)) or a differential operator (as in PINNs; e.g., see Raissi et al. (2019)). The discrepancy between the learned model $\mathcal{A}\mathcal{F}_{\boldsymbol{\theta}}$ and the measurements $\boldsymbol{y}$ is measured using an integral-type data fitting loss $\int_0^t \mathcal{L}(\mathcal{A}\mathcal{F}_{\boldsymbol{\theta}}(s), \boldsymbol{y}(s))ds$, where we assume that the function $\boldsymbol{\theta} \mapsto \mathcal{L}(\mathcal{A}\mathcal{F}_{\boldsymbol{\theta}}(s), \boldsymbol{y}(s))$ is convex. For example, setting $\mathcal{L}(\boldsymbol{a},\boldsymbol{b}) = \sum_{i=1}^m |a_i - b_i|$ or $\mathcal{L}(\boldsymbol{a},\boldsymbol{b}) = \sum_{i=1}^m (a_i - b_i)^2$ yields an L_1 or L_2-squared data fitting loss, respectively. The unknown parameters $\boldsymbol{\theta}$ are then learned by solving the following optimization problem:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^n} \lambda \int_0^t \mathcal{L}(\mathcal{A}\mathcal{F}_{\boldsymbol{\theta}}(s), \boldsymbol{y}(s))ds + R(\boldsymbol{\theta}), \tag{4}$$

where $\lambda > 0$ is a weight on the data fitting loss and R is a convex regularization term.

Then, the connection between the learning problem (4), the generalized Hopf formula (2), and the optimal control problem (3) is summarized in Figure 2. Specifically, the learning problem (4) is related to the generalized Hopf formula (2) by setting $\boldsymbol{\theta} = \mathbf{p}$, $H(s,\mathbf{p}) = \lambda\mathcal{L}(\mathcal{A}\mathcal{F}_{\mathbf{p}}(s), \boldsymbol{y}(s))$, and $R(\mathbf{p}) = J^*(\mathbf{p}) - \langle \mathbf{x}, \mathbf{p}\rangle + c(\mathbf{x})$, where $c(\mathbf{x})$ is a constant (possibly 0) that is independent of $\mathbf{p}$ but may depend on $\mathbf{x}$. In other words, the variable $\mathbf{x}$ in the HJ PDE becomes a hyper-parameter in the learning problem, and we can treat it as a constant when optimizing the learning problem (4) with respect to $\boldsymbol{\theta} = \mathbf{p}$. Hence, when we solve these learning problems, we also solve the optimal control problem (3) with initial position $\mathbf{x}$ and time horizon t, or, equivalently, we evaluate the solution to the corresponding HJ PDE (1) at the point $(\mathbf{x}, t)$. Conversely, when we solve the HJ PDE (1), the spatial gradient $\nabla_{\mathbf{x}} S(\mathbf{x},t)$ of the solution gives the minimizer $\boldsymbol{\theta}^*$ of the learning problem (4).

3. Regularized linear regression problems with quadratic integral-type losses

As a first exploration of the theoretical connection we developed in Section 2.2, we consider the specific case of quadratic-regularized linear regression problems with quadratic integral-type losses.

We begin by establishing the connection for this specific case. We then leverage this connection to develop a new Riccati-based approach for solving this learning problem, which has potential computational and memory advantages in continual learning and big data settings.

3.1. Problem formulation and connection to LQR problems

The learning problem is formulated as follows. Given data $(s, \boldsymbol{y}(s)) \in [0,t] \times \mathbb{R}^m$, the goal is to learn a linear prediction model $\Phi(\cdot)\boldsymbol{\theta}$ such that $\mathcal{A}\Phi(s)\boldsymbol{\theta} \approx \boldsymbol{y}(s), \forall s \in [0,t]$, where $\Phi(\cdot) = [\phi_1(\cdot), ..., \phi_n(\cdot)] \in \mathbb{R}^{m\times n}$ is the matrix whose columns are the basis functions $\phi_j : [0,t] \to \mathbb{R}^m$, $j = 1, \ldots, n$, $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^T \in \mathbb{R}^n$ are unknown trainable coefficients, and $\mathcal{A}$ is a linear operator (e.g., identity operator, differential operators), where we abuse notation and denote $\mathcal{A}\Phi(s) := [\mathcal{A}\phi_1(s), ..., \mathcal{A}\phi_n(s)]$. We learn $\boldsymbol{\theta}$ by minimizing the following loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \int_0^t \frac{\lambda}{2} \|\mathcal{A}\Phi(s)\boldsymbol{\theta} - \boldsymbol{y}(s)\|_2^2 ds + \frac{1}{2}\|A\boldsymbol{\theta} - \boldsymbol{b}\|_2^2, \tag{5}$$

where $\frac{1}{2}\|A\boldsymbol{\theta} - \boldsymbol{b}\|_2^2$ is a quadratic regularization term, $A \in \mathbb{R}^{n\times n}$ is a positive definite matrix that weights the regularization, and $\boldsymbol{b} \in \mathbb{R}^n$ acts as a prior on $\boldsymbol{\theta}$ that biases $\boldsymbol{\theta}$ to be close to $A^{-1}\boldsymbol{b}$. Note that this loss function (5) is strictly convex; hence, it has a unique global minimizer.

Then, this learning problem has connections with a generalized Hopf formula and an optimal control problem. By expanding the square, we see that computing the minimizer of the loss function (5) is equivalent to computing the maximizer of the following generalized Hopf formula:

$$S(\mathbf{x}, t) = \sup_{\mathbf{p}\in\mathbb{R}^n} \left\{ \langle \mathbf{x}, \mathbf{p}\rangle - \int_0^t \frac{\lambda}{2} \|\mathcal{A}\Phi(s)\mathbf{p} - \boldsymbol{y}(s)\|_2^2 ds - \frac{1}{2}\|A\mathbf{p}\|_2^2 + \langle A\mathbf{p}, \boldsymbol{b}\rangle \right\}, \tag{6}$$

when $\mathbf{x} = 0$. In other words, solving this learning problem (5) is equivalent to solving the HJ PDE (1) with time-dependent, quadratic Hamiltonian $H(s, \mathbf{p}) = \frac{\lambda}{2}\|\mathcal{A}\Phi(s)\mathbf{p} - \boldsymbol{y}(s)\|_2^2$ and quadratic initial condition $J(\mathbf{x}) = \frac{1}{2}\|(A^{-1})^T\mathbf{x} + \boldsymbol{b}\|_2^2$ at the point $(0, t)$. By extension, solving both of these problems is also equivalent to solving the optimal control problem (3) with running cost $L(s, \mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathbf{u} - \sqrt{\lambda}\boldsymbol{y}(s)^T\mathbf{u}$, terminal cost $J(\mathbf{x}) = \frac{1}{2}\|(A^{-1})^T\mathbf{x} + \boldsymbol{b}\|_2^2$, dynamics $f(s, \mathbf{u}) = \sqrt{\lambda}(\mathcal{A}\Phi(s))^T\mathbf{u}$, initial position $\boldsymbol{x}(0) = 0$, and time horizon t. Note that since this optimal control problem has quadratic running and terminal costs and linear dynamics, it is actually an LQR problem with time-dependent costs and dynamics (e.g., see Anderson and Moore (2007); Tedrake (2023)).

3.2. Riccati-based methodology

It is well-known that LQR problems can be solved using Riccati ODEs (e.g., see McEneaney (2006)). By our connection, this gives us that the learning problem (5) can also be solved using Riccati ODEs. Namely, the viscosity solution S (given by (6)) to the corresponding HJ PDE is also given by $S(\mathbf{x}, t) = \frac{1}{2}\mathbf{x}^T P(t)\mathbf{x} + \mathbf{q}(t)^T\mathbf{x} + r(t)$, where $P : [0,\infty) \to \mathbb{R}^{n\times n}$, which takes values in the space of positive definite matrices, $\mathbf{q} : [0,\infty) \to \mathbb{R}^n$, and $r : [0,\infty) \to \mathbb{R}$ satisfy the following Riccati ODEs:

$$\begin{cases} \dot{P}(s) = -\lambda P(s)^T(\mathcal{A}\Phi(s))^T\mathcal{A}\Phi(s)P(s) & s > 0, \\ \dot{\mathbf{q}}(s) = -\lambda P(s)^T(\mathcal{A}\Phi(s))^T(\mathcal{A}\Phi(s)\mathbf{q}(s) - \boldsymbol{y}(s)) & s > 0, \\ \dot{r}(s) = -\dfrac{\lambda}{2}\|\mathcal{A}\Phi(s)\mathbf{q}(s) - \boldsymbol{y}(s)\|_2^2 & s > 0 \end{cases} \tag{7}$$

with initial conditions $P(0) = A^{-1}(A^{-1})^T$, $\mathbf{q}(0) = A^{-1}\boldsymbol{b}$, and $r(0) = \frac{1}{2}\boldsymbol{b}^T\boldsymbol{b}$. Then, the minimizer $\boldsymbol{\theta}^*$ of the learning problem (5) is given by $\boldsymbol{\theta}^* = \mathbf{p}^* = \nabla_{\mathbf{x}} S(0, t) = \mathbf{q}(t)$, where $\mathbf{p}^*$ is the optimizer in the generalized Hopf formula (6) when $\mathbf{x} = 0$.

Remark 1 *When solving the learning problem (5), we only care about computing the minimizer $\boldsymbol{\theta}^*$ and not the minimal value of the loss function. Hence, solving the learning problem using the Riccati ODEs (7) only requires solving the ODEs for $P, \mathbf{q}$, and the ODE for r can be ignored. In the remainder of the paper, we disregard the ODE for r.*

Solving the learning problem (5) via the Riccati ODEs (7) provides computational and memory advantages in certain learning scenarios, particularly those involving continual learning and/or very large datasets. For example, consider the case where our model has already been trained using the data $(s, \boldsymbol{y}(s)) \in [0, t] \times \mathbb{R}^m$ and we want to incorporate some additional data points $(s, \boldsymbol{y}(s)) \in (t, t_1] \times \mathbb{R}^m$, where $t_1 > t$, into our learned model. Then, our model can be updated by simply evolving the Riccati ODEs (7) from t to t_1. Note that doing so requires neither retraining on the entire dataset $(s, \boldsymbol{y}(s)) \in [0, t_1] \times \mathbb{R}^m$ nor access to any of the previous data $(s, \boldsymbol{y}(s)) \in [0, t] \times \mathbb{R}^m$ since evolving the Riccati ODEs (7) from t to t_1 only requires the values $P(t), \mathbf{q}(t)$ (i.e., the results of the previous training) to initialize the Riccati ODEs and the new data $(s, \boldsymbol{y}(s)) \in (t, t_1] \times \mathbb{R}^m$ to continue evolving the ODEs on the subsequent time interval $[t, t_1]$. Hence, this Riccati-based approach is particularly beneficial in continual learning scenarios, wherein learned models are incrementally updated as new data becomes available (and hence, constantly retraining on the entire, growing dataset can quickly become computationally infeasible), and memory-constrained learning scenarios where datasets are too large to store in their entirety.

Now consider the case where we want to remove some data $(s, \boldsymbol{y}(s)) \in [t_2, t] \times \mathbb{R}^m$, where $t_2 < t$, which has already been incorporated into our learned model. This scenario may correspond to the case where we want to remove some corrupted data in order to improve the accuracy of the learned model. Then, this data can be removed by reversing time and solving the Riccati ODEs (7) from t to t_2 with terminal conditions given by $P(t), \mathbf{q}(t)$. As in the previous case, the advantages of this Riccati-based approach are that we only require access to the data that is being removed and not the entire dataset. Thus, if we are removing a small amount of data (relative to the size of the entire dataset), this approach can be more efficient than retraining on the entire remaining dataset.

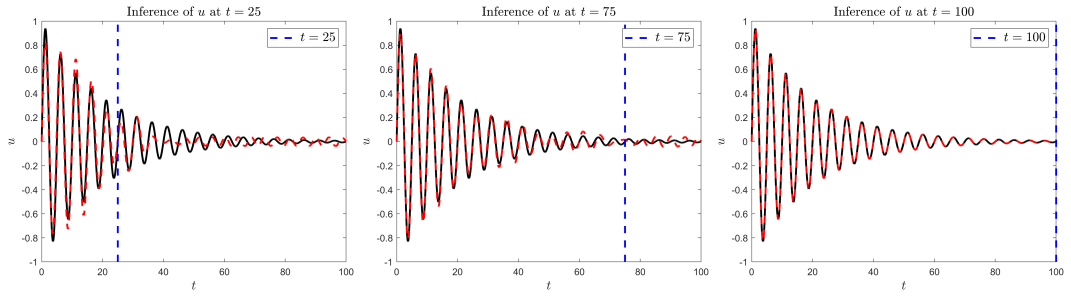
Remark 2 *We have flexibility in how we connect the learning problem (5) to an HJ PDE. Specifically, in (6), we interpret $\boldsymbol{b}$ as part of the initial condition J of the corresponding HJ PDE. However, we could also interpret $\boldsymbol{b}$ to be part of $\mathbf{x}$, the point at which we evaluate the HJ PDE. Let $\boldsymbol{b} = \boldsymbol{b}_J + \boldsymbol{b}_{\mathbf{x}}$, where $\boldsymbol{b}_J$ and $\boldsymbol{b}_{\mathbf{x}}$ are the parts of $\boldsymbol{b}$ that will be associated with J and $\mathbf{x}$, respectively. Then, the new corresponding HJ PDE and optimal control problem are identical to those defined in Section 3.1 but with $\boldsymbol{b}$ replaced with $\boldsymbol{b}_J$ and evaluated at $\mathbf{x} = \boldsymbol{x}(0) = A^T \boldsymbol{b}_{\mathbf{x}}$ (instead of at $\mathbf{x} = \boldsymbol{x}(0) = 0$). Let $\tilde{S}$ be the solution to this new HJ PDE. Then, the minimizer $\boldsymbol{\theta}^*$ of the learning problem (5) can alternatively be computed as $\boldsymbol{\theta}^* = \mathbf{p}^* = \nabla_{\mathbf{x}} \tilde{S}(A^T \boldsymbol{b}_{\mathbf{x}}, t) = \tilde{P}(t) A^T \boldsymbol{b}_{\mathbf{x}} + \tilde{\mathbf{q}}(t)$, where $\tilde{P}, \tilde{\mathbf{q}}$ satisfy the Riccati ODEs (7) with initial conditions $P(0) = A^{-1}(A^{-1})^T$, $\mathbf{q}(0) = A^{-1} \boldsymbol{b}_J$. The advantage of this alternative interpretation is that $\boldsymbol{b}_{\mathbf{x}}$ (i.e., the bias on $\boldsymbol{\theta}$) can be tuned efficiently using only 2 matrix-vector multiplications and 1 vector addition instead of having to re-solve the Riccati ODEs (7) with a new initial condition (i.e., re-training on the entire dataset with a new bias). Note that when $\boldsymbol{b}_{\mathbf{x}} = 0$, we recover the original framework from Section 3.1.*

4. Numerical examples

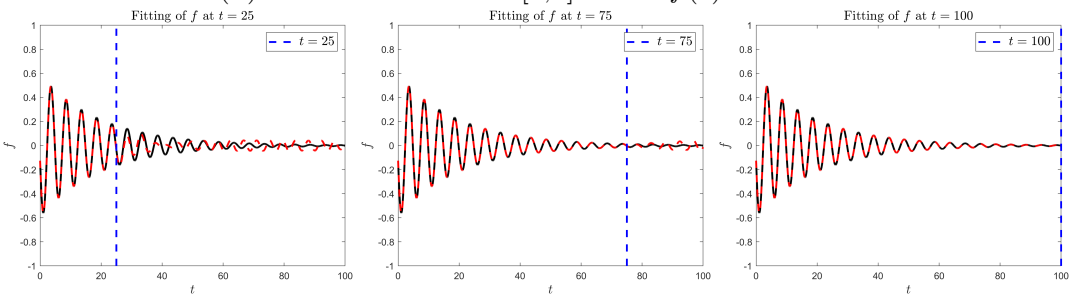
In this section, we apply the Riccati-based methodology from Section 3 to two test problems to demonstrate the potential computational and memory advantages of our approach. For illustration purposes, we apply 4th-order Runge-Kutta (RK4) to solve the Riccati ODEs in each example. Note that our methodology does not rely on any particular numerical solver. Hence RK4 could be

replaced by any other appropriate numerical method. Code for all examples will be made publicly available at https://github.com/ZongrenZou/TimeHJPDE4SciML after this paper is accepted.

4.1. A boundary-value ODE problem



(a) Inference of u after $[0, t] \ni s \mapsto f(s)$ is visited.



(b) Inference/fitting of f after $[0, t] \ni s \mapsto f(s)$ is visited.

Figure 3: Continual learning of the solution u and source term f of the boundary-value ODE (8) using our Riccati-based approach, where information of f is treated as a flow with respect to t and cannot be stored once visited. −−: inferences of u, f at different t; —: exact values of u, f; −−: where integration has advanced in t so far. Our Riccati-based approach naturally coincides with the continual learning framework by allowing new information to be continuously incorporated into the learned model without requiring access to any of the previous information. Instead, all of the previous information is encoded in the solution to the corresponding HJ PDE, thus avoiding catastrophic forgetting.

In this example, we apply our Riccati-based methodology to solve a boundary value problem using continual learning (Parisi et al. (2019); Van de Ven and Tolias (2018)) to demonstrate the potential computational and memory benefits of our approach. Consider the following ODE problem:

$$\begin{cases} \dfrac{d^2u}{dt^2} + u(t) = f(t), t \in (0, T), \\ u(0) = u_0, u(T) = u_T, \end{cases} \tag{8}$$

where $f : [0, T] \to \mathbb{R}$ is the source term derived from $u(t) = \exp(-0.05t)\sin(0.4\pi t)$, $t \in [0, T]$ (although u is assumed to be unknown a priori), $u_0 = 0$ and $u_T = \exp(-0.05T)\sin(0.4\pi T)$ are constants, and $T = 100$ so that this is a long-term integration problem. Following the continual learning framework, we assume that information of f is accessed in a stream as a flow of t and that information of f becomes inaccessible after it is incorporated into our learned model. Our goal is to learn the linear model $\sum_{i=1}^{n} \theta_i \phi_i(\cdot)$ to approximate the solution u of the ODE (8),

where $\{\phi_i(\cdot)\}_{i=1}^n = \{1\} \cup \{j\sin(j\cdot), j\cos(j\cdot)\}_{j=1}^{\frac{n-1}{2}}, n = 301$. We learn the unknown coefficients $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^T$ using the following PINN-type loss (Raissi et al. (2019); Zou et al. (2024)):

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}; t) = \frac{\lambda_f}{2}\int_0^t \left|\sum_{i=1}^n \theta_i \left(\frac{d^2\phi_i}{dt^2}(s) + \phi_i(s)\right) - f(s)\right|^2 ds \\ + \frac{\lambda_0}{2}\left|\sum_{i=1}^n \phi_i(0)\theta_i - u_0\right|^2 + \frac{\lambda_T}{2}\left|\sum_{i=1}^n \phi_i(T)\theta_i - u_T\right|^2 + \frac{1}{2}\sum_{i=1}^n \gamma_i|\theta_i|^2,\end{aligned} \tag{9}$$

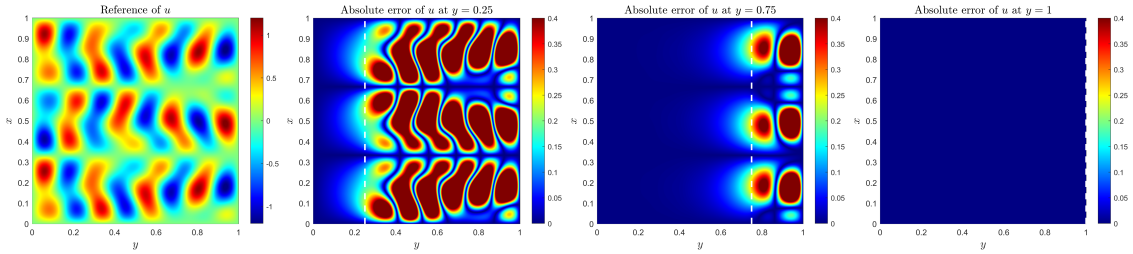
where $\lambda_f > 0$ is the belief weight for the ODE, $\lambda_0, \lambda_T \geq 0$ are belief weights for the boundary conditions, and $\gamma_i > 0, i = 1, \ldots, n$ are belief weights for the ℓ^2-regularization. Following the data streaming paradigm, to achieve real-time inferences, we need to minimize $\mathcal{L}(\cdot; t)$ for all $t \in [0, T]$. Note that by completing the square, minimizing the above loss function (9) is equivalent to minimizing a loss function in the form of (5), where $\lambda = \lambda_f$, $\Phi(\cdot) = [\phi_1(\cdot), \ldots, \phi_n(\cdot)]$, $\mathcal{A}\Phi(\cdot) = [\frac{d^2\phi_1}{dt^2}(\cdot) + \phi_1(\cdot), \ldots, \frac{d^2\phi_n}{dt^2}(\cdot) + \phi_n(\cdot)]$, $\boldsymbol{y} = f$, $A = (\lambda_0\Phi(0)^T\Phi(0) + \lambda_T\Phi(T)^T\Phi(T) + \Gamma)^{1/2}$, where Γ is the diagonal matrix whose i-th entry is γ_i, and $\boldsymbol{b} = A^{-1}(\lambda_0\Phi(0)^T u_0 + \lambda_T\Phi(T)^T u_T)$. In other words, we treat the sum of the boundary and regularization terms in (9) as the regularization term in (5). Thus, the learning problem (9) can be solved using the Riccati ODEs (7).

For our numerics, we use $\lambda_f = 100, \lambda_0 = \lambda_T = \gamma_i = 1, i = 1, ..., n$, and RK4 with step size $h = 10^{-6}$. Note that evolving the Riccati ODEs from t to $t+h$ only requires information of $f(s), s \in [t, t+h]$. Hence, our approach naturally coincides with the continual learning framework. In Figure 3, we observe that our inferences of both u and f improve as more information is incorporated into our learned model. In particular, we see that our inferences improve in accuracy in both the regions we have already visited and at future times, which indicates that catastrophic forgetting has not occurred. This behavior is consistent with the fact that all of the information from $s < t$ is inherently encoded in the solution to the corresponding HJ PDE via the solutions $P(t), \mathbf{q}(t)$ to the Riccati ODEs. Specifically, the relative L_2 errors of the inferences of u are 27.13%, 12.36%, 0.03% once $t = 25, 75, 100$ has been visited, respectively. Similarly, the relative L_2 errors of the inferences of f are 23.83%, 8.08%, 0.03% once $t = 25, 75, 100$ has been visited, respectively. For comparison, we also compute the least squares estimate (LSE) for (9), in which the integral is approximated using Monte Carlo with 10^6 uniform sampling points. The relative L_2 errors of the LSE inferences of u and f are both 0.03%. We compute all of the above L_2 errors using trapezoidal rule with a uniform grid of size 1001 over the whole domain $[0, 100]$. Note that LSE does not meet the requirements of continual learning as it requires access to the full information of f at once (i.e., LSE requires information of $f(t), \forall t \in [0, 100]$), yet our approach yields similar error levels once $t = 100$ has been visited even though we only access a small portion of the data at a time. We also note that the inferences made using our approach can be considered in real time as they are updated every h time units of data. Thus, our approach provides potential computational and memory advantages as updating the learned model using information on $[t, t+h]$ does not require retraining on or storage of any of the previous information from $[0, t)$.

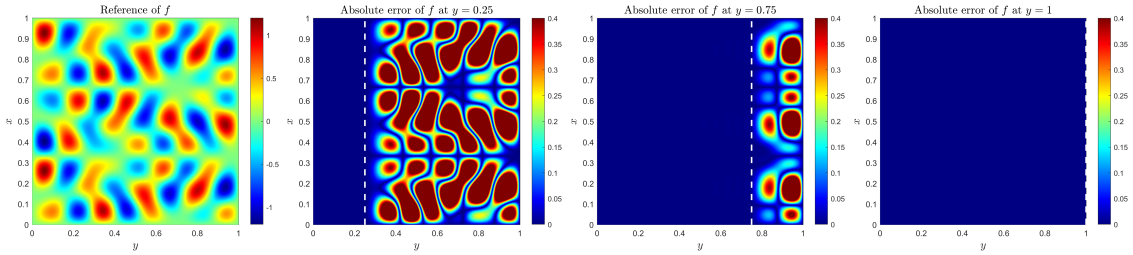
4.2. 2D Poisson equation

In this example, we extend our Riccati-based approach to a higher dimensional problem to demonstrate the potential memory advantages of our approach. Consider the 2D Poisson equation:

$$\begin{cases} -\kappa\Delta u(x, y) = f(x, y), & (x, y) \in (\Omega \setminus \partial\Omega), \\ u(x, y) = 0, & (x, y) \in \partial\Omega, \end{cases} \tag{10}$$



(a) Absolute errors of the inference of u after $[0,1] \times [0,y] \ni (x,t) \mapsto f(x,t)$ is visited.



(b) Absolute errors of the inference/fitting of f after $[0,1] \times [0,y] \ni (x,t) \mapsto f(x,t)$ is visited.

Figure 4: Continual learning of the solution u and source term f of the 2D Poisson equation (10) using our Riccati-based approach. White dashed lines: where integration has advanced in y so far. Information of f is discretized in x and then propagated along y. Hence, our approach only requires access to 1D slices of the domain instead of the entire domain, which highlights the potential memory benefits of our Riccati-based approach.

where $\Omega := [0,1]^2$ is the domain, $\kappa = \frac{0.01}{\pi^2}$ is a constant, and $f : \Omega \to \mathbb{R}$ is the source term. Assume that we only have access to a few slices of f due to limited computational resources, which prevents direct deployment of most traditional numerical solvers, which typically require full information of f to solve the equation either iteratively or in one-shot. Hence, we can solve this PDE using continual learning to take advantage of the memory benefits of data streaming. Namely, we will avoid having to store the entire discretized grid of Ω by only processing/accessing information on a small portion of Ω at any given time. We learn a linear model $(x,y) \mapsto \sum_{i=1}^n \theta_i \phi_i(x,y)$ to approximate the solution u, where the coefficients $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^T$ are learned by minimizing:

$$\mathcal{L}(\boldsymbol{\theta}; t) = \frac{\lambda_f}{2N} \int_0^t \sum_{j=1}^{N+1} \left[\kappa \sum_{i=1}^n \theta_i \left(\frac{\partial^2 \phi_i}{\partial x^2} + \frac{\partial^2 \phi_i}{\partial y^2} \right)(x_j, y) + f(x_j, y) \right]^2 dy + \frac{1}{2} \sum_{i=1}^n \gamma_i |\theta_i|^2, \tag{11}$$

where λ_f and $\gamma_i, i = 1, ..., n$ are belief weights. Note that we omit the boundary loss in (11) since the boundary conditions will be enforced by our choice of $\{\phi_i\}_{i=1}^n$. We also have discretized the integral-type loss in x with a uniform mesh $x_j = \frac{j-1}{N}, j = 1, \ldots, N+1$, so that the integration along y can be treated as the evolution in "time" of the corresponding HJ PDE. The above loss function (11) is in the same form of (5) by setting $\lambda = \frac{\lambda_f}{N}$, $[\Phi(\cdot)]_{ji} = \phi_i(x_j, \cdot)$, $[\mathcal{A}\Phi(\cdot)]_{ji} = \kappa(\frac{\partial^2 \phi_i}{\partial x^2} + \frac{\partial^2 \phi_i}{\partial y^2})(x_j, \cdot)$, $\boldsymbol{y}(\cdot) = -[f(x_1, \cdot), \ldots, f(x_{N+1}, \cdot)]^T$, $A = \Gamma^{1/2}$, where Γ is a diagonal matrix whose i-th entry is γ_i, and $\boldsymbol{b} = 0$. Thus, this learning problem (11) can be solved using the Riccati ODEs (7) to perform the integration along y, and instead of storing information of f on all of Ω, we only require information along 1D slices of Ω at any given time, which provides memory advantages over more traditional discretization methods.

In our numerical experiments, we consider the case where f is derived from

$$u(x,y) = -0.8\sin(3\pi x)\sin(8\pi y) + 0.4\sin(9\pi x)\sin(7\pi y) - 0.3\sin(6\pi x)\sin(10\pi y)$$

(but u is assumed to be unknown a priori), $N = 400$, $\lambda = 100$, $\gamma_i = 1, i = 1, ..., n$, $\{\phi_i\}_{i=1}^{m^2} = \{(x,y) \mapsto \sin(j\pi x)\sin(k\pi y)\}_{j,k=1}^{m}$, $m = 15$ ($n = m^2$), and we employ RK4 with step size $h = 10^{-5}$. We also assume that our measurements of f are corrupted with additive Gaussian noise with mean zero and scale 0.05, which justifies the use of ℓ^2-regularization. In Figure 4, we see that our Riccati-based approach obtains increasingly accurate inferences of f and u as more noisy data is incorporated into our learned models. Specifically, the relative L_2 errors of the inferences of u are 81.42%, 34.00%, 0.07% and of f are 81.87%, 30.30%, 0.03% once $y = 0.25, 0.75, 1$ has been visited, respectively. For comparison, we also compute the LSE for (11), in which the integral is approximated using Monte Carlo with 10^4 uniform sampling points. The relative L_2 errors of the LSE inferences of u and f are 0.30% and 0.09%, respectively. We compute all of the above L_2 errors using trapezoidal rule with a uniform 401×401 grid over all of Ω. Note that LSE requires information of all of Ω at once, yet our approach obtains more accurate inferences once all of Ω has been visited despite only accessing information along one 1D slice of Ω at a time. The accuracy of the LSE inferences could be improved by increasing the number of Monte Carlo points. However, the LSE with 10^5 Monte Carlo points is not able to be computed on a standard laptop (13th Gen Intel(R) Core(TM) i9-13900HX with 2.20 GHz processor and 16 GB RAM) due to memory constraints, which highlights the memory advantages of our approach. In this case, LSE would require an $N \times 10^5$ grid of Ω, whereas our approach only requires N points at a time, where the accuracy in y can be improved without any additional memory burden by decreasing h.

5. Summary

In this paper, we established a new theoretical connection between regularized learning problems with integral-type losses and the generalized Hopf formula for HJ PDEs with time-dependent Hamiltonians. This connection yields a new interpretation for certain SciML applications. Namely, when we solve these learning problems, we also solve an optimal control problem and its associated HJ PDE with time-dependent Hamiltonian. In the special case of linear regression, we leveraged this connection to develop a new Riccati-based methodology that provides promising computational and memory advantages in continual learning settings, while inherently avoiding catastrophic forgetting.

This work opens opportunities for many exciting future directions. For example, in contrast to the methodology in Chen et al. (2024a), our methodology (Section 3.2) does not allow data to be added or removed in arbitrary order since the corresponding HJ PDE must be evolved continuously in time. Achieving the same flexibility as the methodology in Chen et al. (2024a) requires relaxing the convexity and regularity assumptions for the generalized Hopf formula in Section 2. Thus, a worthwhile future direction of this work would be to extend to nonconvex and/or discontinuous Hamiltonians, which, in turn, would extend our connection to be between more general learning problems (e.g., those with nonconvex loss functions) and differential games (Evans and Souganidis (1984)), instead of optimal control. In Section 4.2, we extended our Riccati-based approach to a 2D example, and as an illustration we propagated the information along the y-axis. For more general problems, it would be valuable to perform a more in-depth investigation into selecting an appropriate propagation direction, so that the memory load of this approach remains sufficiently reduced in higher dimensions. Another natural extension would be to explore how our connection could be leveraged to reuse existing efficient machine learning algorithms to solve high-dimensional HJ PDEs and optimal control problems as so far, we have only explored the opposite direction.

Acknowledgments

P.C. is funded by the Office of Naval Research (ONR) In-House Laboratory Independent Research Program (ILIR) at NAWCWD managed by Dr. Claresta Dennis and a Department of Defense (DoD) SMART Scholarship for Service SEED Grant. (Approved for public release. Distribution is unlimited. PR 24-0095.) Z.Z., J.D., and G.E.K. are supported by the DOE-MMICS SEA-CROGS project (DE-SC0023191) and the MURI/AFOSR project (FA9550-20-1-0358). G.E.K. is also supported by the ONR Vannevar Bush Faculty Fellowship (N00014-22-1-2795).

References

Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

Martino Bardi and Italo Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Systems & Control: Foundations & Applications. Birkhäuser Boston, Inc., Boston, MA, 1997. ISBN 0-8176-3640-4. doi: 10.1007/978-0-8176-4755-1. URL https://doi.org/10.1007/978-0-8176-4755-1. With appendices by Maurizio Falcone and Pierpaolo Soravia.

Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

Paula Chen, Tingwei Meng, Zongren Zou, Jérôme Darbon, and George Em Karniadakis. Leveraging Multi-time Hamilton-Jacobi PDEs for Certain Scientific Machine Learning Problems. *SIAM Journal on Scientific Computing*, 46:C216–C248, 2024a.

Paula Chen, Tingwei Meng, Zongren Zou, Jérôme Darbon, and George Em Karniadakis. Leveraging Hamilton-Jacobi PDEs with time-dependent Hamiltonians for continual scientific machine learning. *arXiv preprint arXiv:2311.07790*, 2024b.

Jérôme Darbon and Tingwei Meng. On decomposition models in imaging sciences and multi-time Hamilton–Jacobi partial differential equations. *SIAM Journal on Imaging Sciences*, 13(2):971–1014, 2020. doi: 10.1137/19M1266332. URL https://doi.org/10.1137/19M1266332.

Lawrence C. Evans and Panagiotis E. Souganidis. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations. *Indiana University Mathematics Journal*, 33 (5):773–797, 1984.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

P. L. Lions and J-C. Rochet. Hopf Formula and Multitime Hamilton-Jacobi Equations. *Proceedings of the American Mathematical Society*, 96(1):79–84, 1986. URL http://www.jstor.org/stable/2045657.

W. McEneaney. *Max-plus methods for nonlinear control and estimation*. Springer Science & Business Media, 2006.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Maziar Raissi, Paris Perdikaris, and George 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.

J-C. Rochet. The taxation principle and multi-time Hamilton-Jacobi equations. *Journal of Mathematical Economics*, 14:113–128, 1985.

Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

Russ Tedrake. *Underactuated Robotics*. Course Notes for MIT 6.832, 2023. URL https://underactuated.csail.mit.edu.

Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. NeurIPS, Continual Learning Workshop, 2018.

Sanford Weisberg. *Applied linear regression*, volume 528. John Wiley & Sons, 2005.

Zongren Zou, Xuhui Meng, and George Em Karniadakis. Correcting model misspecification in physics-informed neural networks (PINNs). *Journal of Computational Physics*, 505, 2024.