Neural Time-Reversed Generalized Riccati Equation

Alessandro Betti[1], Michele Casoni[2], Marco Gori[1, 2], Simone Marullo[2, 3],
Stefano Melacci[2], Matteo Tiezzi[2]

[1] Inria, Lab I3S, MAASAI, Universitè Côte d'Azur, Nice, France,
[2] DIISM, University of Siena, Siena, Italy,
[3] DINFO, University of Florence, Florence, Italy
alessandro.betti@inria.fr, m.casoni@student.unisi.it, {marco,mela, mtiezzi}@diism.unisi.it, simone.marullo@unifi.it

Abstract

Optimal control deals with optimization problems in which variables steer a dynamical system, and its outcome contributes to the objective function. Two classical approaches to solving these problems are *Dynamic Programming* and the *Pontryagin Maximum Principle*. In both approaches, Hamiltonian equations offer an interpretation of optimality through auxiliary variables known as *costates*. However, Hamiltonian equations are rarely used due to their reliance on forward-backward algorithms across the entire temporal domain. This paper introduces a novel neural-based approach to optimal control, with the aim of working forward-in-time. Neural networks are employed not only for implementing state dynamics but also for estimating costate variables. The parameters of the latter network are determined at each time step using a newly introduced *local* policy referred to as the *time-reversed generalized Riccati equation*. This policy is inspired by a result discussed in the Linear Quadratic (LQ) problem, which we conjecture stabilizes state dynamics. We support this conjecture by discussing experimental results from a range of optimal control case studies.

Introduction

Optimal control (Lewis, Vrabie, and Syrmos 2012) offers a wide framework to set up optimization problems that are concerned with the steering of a dynamical system in some parsimonious way. It is therefore clear that its scope is quite large and it intersects many areas such as, for instance, pure math, natural sciences and engineering. Being the optimization problem objective defined on the solution of a system of ODEs over a certain temporal horizon $[t_0, T]$, it has a global-in-time nature. Indeed, classical approaches to optimal control such as the *Pontryagin Maximum Principle* (see (Gamkrelidze, Pontrjagin, and Boltjanskij 1964; Giaquinta and Hildebrandt 2013)) and *dynamic programming* (see (Bardi, Dolcetta et al. 1997)) both characterize solutions in terms of a boundary problem for some differential conditions (usually a PDE in dynamic programming and a system of ODEs with the Pontryagin maximum principle). This means that, in general, the algorithms to find solutions require iterative forward/backward approaches to glue the local-in-time computations of the differential equations with the boundary conditions at opposite sides of the temporal interval.

In many instances of control problems, where either the complexity of the model is high and/or the temporal horizon could be very long, as it could happen for instance in Reinforcement Learning (Sutton and Barto 2018; Bertsekas 2019) or Lifelong Learning (Betti et al. 2022; Mai et al. 2022), these methods are unfeasible and we usually need to resort to different control strategies. A typical approach is that of using *Model Predictive Control* (Garcia, Prett, and Morari 1989) (also known as *receding horizon control*), with a real-time iteration (RTI) scheme for solving the online optimization problem (Diehl, Bock, and Schlöder 2005). The necessity of finding optimization procedures that only exploit forward (in time) computations is an especially sensible matter within the machine learning community, where the possibility of performing a backpropagation through the entire temporal horizon (backpropagation through time) is considered to be extremely implausible from a biological point of view (Hinton 2022) and in some cases prohibitively costly.

In this work we present a novel approach that makes use of Hamilton Equations giving an estimate of the costate function through a neural-network computation, working *forward-in-time*. The basic idea of our approach is to estimate the parameters of this network by means on an indirect usage of the Hamilton equations. Recently, in (Jin et al. 2019), the possibility of exploiting Hamiltonian equations for learning system dynamics and controlling policies forward in time has been investigated. In this paper, the authors introduced *Pontryagin Differentiable Programming* (PDP) to efficiently compute the gradients of the state trajectory with respect to the system parameters using an auxiliary control system. This approach differs from the method proposed in this paper by the fact that, instead, in the present work, we use Hamilton equations indirectly for defining an optimization problem for the temporal variations of the model parameters. In doing so, we are basically defining a dynamic on the parameters that estimate the costate in a similar manner as we would do with the Riccati equation in the Linear Quadratic control problem. We conjecture that the resulting time-reversed dynamics will lead to a stabilizing effect on the state equation, hence opening the possibility to use this method forward-in-time.

This approach has been inspired by the possibility of using optimal control techniques in the continual online learning scenario recently proposed in (Betti et al. 2022) to formulate a class of lifelong problems using the formalism of control theory. For this reason, throughout the paper we assume that the dynamical system that defines the evolution of the state is also expressed by a neural model, in the form of a continuous time recurrent neural network (Zhang, Wang, and Liu 2014). The authors in (Betti et al. 2022) proposed a method to enforce stability by pushing the costate dynamic to converge to zero, and hence directly interfering with the dynamics prescribed by Hamilton equations. Conversely, we directly leverage on Hamilton equations to devise a stabilizing policy for the system.

The paper is organized as follows. In Section we describe the class of dynamical models that we take into account, Section is devoted to the formulation of the control problem and contains a short review of the main results from optimal control that will be used in the reminder of the paper. Section constitutes the core part of the contribution and it is where we introduce the time-reversed generalized Riccati equation. Section contains the experimental observations that have been organized in three different case studies, while conclusions and ideas for future work are the subjects of Section .

Continuous Time State Model

Let us focus on models that depend on N parameters, whose values at time t are yielded by $\alpha(t)$, and that are based on an internal state $x(t)$ of size n which dynamically changes over time. We consider a classic state model

$$x'(t) = f(x(t), \alpha(t), t), \quad t \in (t_0, T] \qquad (1)$$

where $f \colon \mathbb{R}^n \times \mathbb{R}^N \times [t_0, T] \to \mathbb{R}^n$ is a Lipschitz function, $t \mapsto \alpha(t)$ is the trajectory of the parameters of the model, which is assumed to be a *measurable function*, and T is the temporal horizon on which the model is defined; $t_0 \geq 0$. We assume that the p-components output of the model is computed by a *fixed* transformation of the state, $\pi \colon \mathbb{R}^n \to \mathbb{R}^p$, usually a projection of class $C^\infty(\mathbb{R}^n; \mathbb{R}^p)$. The initial state of the model is assigned to a fixed vector $x^0 \in \mathbb{R}^n$, that is

$$x(t_0) = x^0. \qquad (2)$$

Let us now pose $\mathcal{A} := \{\alpha \colon [t_0, T] \to \mathbb{R}^N : \alpha \text{ is measurable}\}$.

Definition 1. Given a $\beta \in \mathcal{A}$, and given an initial state x^0, we define the *state trajectory*, that we indicate with $t \mapsto x(t; \beta, x^0, t_0)$, the solution of (1) with initial condition (2).

The goal of this work is to define a procedure to estimate with a *forward-in-time* scheme an approximation of the optimal control parameters α.[1] Notice that the explicit time dependence t of Eq. (1) is necessary to take into account the provision over time of some input data to the model. In the next section, we will give a more precise structure to such temporal dependence.

Neural State Model

We implement the function f of Eq. (1) by a neural network γ, where the dependence on time t is indirectly modeled by a novel function $u(t)$, that yields the d-dimensional input data provided at time t to the network. Formally, for all $\xi \in \mathbb{R}^n$, for all $s \in [t_0, T]$ and all $a \in \mathbb{R}^N$, $f(\xi, a, s) := \gamma(\xi, u(s), a)$, where, for all fixed $a \in \mathbb{R}^N$, the map $\gamma(\cdot, \cdot, a) \colon \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ is a neural network and $u \colon [t_0, T] \to \mathbb{R}^d$ is the input signal, being $u \in BV((t_0, T))$ an assigned input map of bounded variation[2]. More directly we can assume that we are dealing with a Continuous Time Recurrent Neural Network (CTRNN) (see (Zhang, Wang, and Liu 2014)) that at each instant estimates the variation of the state based on the current value of the state itself and on an external input. The network $\gamma(\cdot, \cdot, a)$ represents the transition function of the state. In this new notation, the dynamic of the state, given by Eq. (1) together with Eq. (2), is described by the following Cauchy problem for x:

$$\begin{cases} x'(t) = \gamma(x(t), u(t), \alpha(t)), & \text{for } t \in (t_0, T] \\ x(t_0) = x^0. \end{cases} \qquad (3)$$

To help the reader in giving an initial interpretation to the parameters α, at this stage it is enough to assume that α could basically represent the weights and the biases of the network γ. Similarly, the state x could be imagined as the usual state in a CTRNN. However, there are still some steps to take before providing both α and x the exact role we have considered in this paper. First, we need to define the way α participates in an optimization problem, defining a *control problem* whose *control parameters* are α. This will be the main topic of the next Section , where we will start from the generic state model of the beginning of Section , and then cast the descriptions on the neural state model γ—Section . When doing it, we will also reconsider the role of α in the context of the neural network γ, due to some requirements introduced by the optimization procedure over time.

Control Problem

Suppose now that we want to use the model described in Eq. (1) paired with Eq. (2) to solve some task that can be expressed as a minimization problem for a cost functional $\alpha \mapsto C(\alpha)$. We recall the notation $x(t; \alpha, x^0, t_0)$, introduced in Def. (1), to compactly indicate the state x and all its dependencies as a solution of Eq. (1) with initial values (2). The cost functional has the following form:

$$C_{x^0, t_0}(\alpha) := \int_{t_0}^T \ell(\alpha(t), x(t; \alpha, x^0, t_0), t) \, dt, \qquad (4)$$

where $\ell(a, \cdot, s)$ is bounded and Lipschitz $\forall a \in \mathbb{R}^N$ and $\forall s \in [t_0, T]$. The function ℓ is usually called *Lagrangian* and it can be thought as the counterpart of a classic machine-learning loss function in control theory. Because the term $x(t; \alpha, x^0, t_0)$ in Eq. (4) depends on the variables α through

---

[1]The meaning of optimality will be described in details in Section

[2]Here the space $BV(t_0, T)$ is the functional space of functions of bounded variation, see (Ambrosio, Fusco, and Pallara 2000).

the integration of a first-order dynamical system, the problem

$$\min_{\alpha \in \mathcal{A}} C_{x^0, t_0}(\alpha) \qquad (5)$$

is a constrained minimization problem which is usually denoted as *control problem* (Bardi, Dolcetta et al. 1997), assuming that a solution exists.

A classical way to address problem (5) is through dynamic programming and the Hamilton-Jacobi-Bellman equation (Bardi, Dolcetta et al. 1997), that will be the key approach on which we will build the ideas of this paper, paired with some intuitions to yield a forward solution over time. We briefly summarize such classical approach in the following. The first step to address our constrained minimization problem is to define the *value function* or *cost to go*, that is a map $v \colon \mathbb{R}^n \times [t_0, T] \to \mathbb{R}$ defined as

$$v(\xi, s) := \inf_{\alpha \in \mathcal{A}} C_{\xi, s}(\alpha), \quad \forall (\xi, s) \in \mathbb{R}^n \times [t_0, T].$$

The optimality condition of the cost C then translates into an infinitesimal condition (PDE) for the value function v (see (Bardi, Dolcetta et al. 1997)); such result can be more succinctly stated once we define the *Hamiltonian* function $H \colon \mathbb{R}^n \times \mathbb{R}^n \times [t_0, T] \to \mathbb{R}$

$$H(\xi, \rho, s) := \min_{a \in \mathbb{R}^N} \{ \rho \cdot f(\xi, a, s) + \ell(a, \xi, s) \}, \quad (6)$$

being $\cdot$ the dot product. Then the following well-known result holds.

Theorem 1 (Hamilton-Jacobi-Bellman). *Let us assume that D denotes the gradient operator with respect to ξ. Furthermore, let us assume that $v \in C^1(\mathbb{R}^n \times [t_0, T], \mathbb{R})$ and that the minimum of $C_{\xi,s}$, Eq. (5), exists for every $\xi \in \mathbb{R}^n$ and for every $s \in [t_0, T]$. Then v solves the PDE*

$$v_s(\xi, s) + H(\xi, Dv(\xi, s), s) = 0, \qquad (7)$$

$(\xi, s) \in \mathbb{R}^n \times [t_0, T)$, *with terminal condition* $v(\xi, T) = 0$, $\forall \xi \in \mathbb{R}^n$. *Equation* (7) *is usually referred to as Hamilton-Jacobi-Bellman equation.*

Proof. See appendix A of (Betti et al. 2023). □

The result stated in Theorem 1 gives a characterization of the value function; the knowledge of the value function in turn gives a direct way to construct a solution of the problem defined in Eq. (5) by a standard procedure called *synthesis procedure* (Evans 2022; Bardi, Dolcetta et al. 1997), for which we summarize its main ingredients. The first step is, once a solution of Eq. (7) with the terminal condition $v(\xi, T) = 0$, $\forall \xi \in \mathbb{R}^n$ is known, to find an *optimal feedback map* $S \colon \mathbb{R}^n \times [t_0, T] \to \mathbb{R}^N$ defined by the condition

$$S(\xi, s) \in \arg\min_{a \in \mathbb{R}^N} \{ Dv(\xi, s) \cdot f(\xi, a, s) + \ell(a, \xi, s) \}. \quad (8)$$

Once a function S with such property is computed, the second step is to solve

$$x'(t) = f(x(t), S(x(t), t), t), \quad \text{for} \quad t \in (t_0, T),$$

with initial condition $x(t_0) = x^0$, and call a solution of this equation x^*. Then the optimal control α^* is directly given by the feedback map:

$$\alpha^*(t) = S(x^*(t), t). \qquad (9)$$

Hamilton Equations There exists another route that can be followed to face the problem of Eq. (5), and that does not directly make use of Hamilton-Jacobi-Bellman equation (7). Such a route, that we will exploit in the rest of the paper, mainly rely on an alternative representation of the value function which is obtained through the *the method of characteristics* (Courant and Hilbert 2008) and which basically makes it possible to compute the solution of Hamilton-Jacobi-Bellman equation along a family of curves that satisfy a set of *ordinary differential equations* (ODEs). This approach is also equivalent (see (Bardi, Dolcetta et al. 1997)) to the Pontryagin Maximum Principle (Giaquinta and Hildebrandt 2013). Let us define the *costate* $p(t) := Dv(x(t), t)$ and consider the following system of ODEs known as *Hamilton Equations*,

$$\begin{cases} x'(t) = H_\rho(x(t), p(t), t); & t \in (t_0, T] \\ p'(t) = -H_\xi(x(t), p(t), t); & t \in (t_0, T] \\ x(t_0) = x^0; \\ p(T) = 0, \end{cases} \qquad (10)$$

being H_ρ and H_ξ the derivatives of H with respect to its second and first argument, respectively. Given a solution to Eq. (10), we can find a solution of Eq. (7) with the appropriate terminal conditions (see (Bardi, Dolcetta et al. 1997)). More importantly, this means that instead of directly finding the value function v, in order to find the optimal control α^* we can solve Eq. (10) to find p^* and x^* and then, as we describe in Eq. (9) and (8), choose

$$\alpha^* \in \arg\min_{a \in \mathcal{A}} \{ p^*(t) \cdot f(x^*(t), a, t) + \ell(a, x^*(t), s) \}. \quad (11)$$

While the problem reformulated in this way appears to be significantly more tractable, having traded a PDE for a system of ODEs, the inherent difficulty of solving a global-in-time optimization problem remains and can be understood as soon as one realizes that Eq. (10) is a problem with both initial and terminal boundary conditions. From a numerical point of view, this means that in general an iterative procedure over the whole temporal interval is needed (for instance shooting methods (Osborne 1969)), making this approach, based on Hamilton equations, unfeasible for a large class of problems when the dimension of the state and/or the length of the temporal interval is big. Finding a forward approach to deal with this issue will be the subject of Section , while our next immediate goal is to bridge the just introduced notions to formalize the role of α in our neural-based approach.

Controlling the Parameters of the Network

Let us now bridge the just described notions with the neural-based implementation γ of the state model, as we have already discussed in Section . In this section, we will give a detailed description of the state variable $x(t)$ associated to the neural computation and we will discuss the specific instance of the controls $\alpha(t)$ we consider, that are both related to the parameters of net γ.

We consider a digraph $G = (V, E)$ and, without loss of generality, let us assume that $V = \{1, \ldots, m\}$. Remember that given a digraph, for each $i \in V$ we can always define

two sets $\text{ch}(i) := \{j \in V : (i,j) \in E\}$ and $\text{pa}(i) := \{j \in V : (j,i) \in E\}$. The digraph becomes a network as soon as we decorate each arc $(j,i) \in E$ with a weight $w_{ij}(t)$ and each vertex $i \in V$ with a neuron output $y_i(t)$ and a bias $b_i(t)$, for every temporal instant $t \in [t_0, T]$. Then the typical CTRNN computation can be written as

$$y_i'(t) = -y_i(t) + \sigma\Big(\sum_{j \in \text{pa}(i)} w_{ij}(t)y_j(t) + b_i(t) \\ + \sum_{j=1}^{d} k_{ij}(t)u_j(t)\Big), \quad (12)$$

where $k_{ij}(t)$ is a component of the weight matrix associated with the input u and $\sigma \colon \mathbb{R} \to \mathbb{R}$ is an activation function. In general, when dealing with optimization over time, we want to be able to impose a regularization not only on the values of the parameters of the network, but also on their temporal variations; as a matter of fact, in many applications one would consider preferable slow variations or, if the optimization fully converged, even constant parameters of the network. For this reason it is convenient to associate the control variables α with the temporal variations (derivatives) of the network's parameters. The classic learnable parameters (weights and biases) of the network can be considered as part of the state x, together with the neuron outputs y. This requires ($i.$) to extend the neural state model of Eq. (3), in order to provide a dynamic to the newly introduced state components, and ($ii.$) to take into account the novel definition of α. Formally, the state at time t becomes $x(t) = (y(t), w(t), b(t), k(t))$ and γ is only responsible of computing the dynamic of the y-portion of it. [3] The state model of Eq. (3), involving all the components of x above, is then,

$$\begin{cases} y'(t) = \gamma(y(t), u(t), w(t), b(t), k(t)) \\ w_{ij}'(t) = \omega_{ij}(t), \quad (j,i) \in E \\ b_i'(t) = \nu_i(t), \quad i \in V \\ k_{ij}'(t) = \chi_{ij}(t), \quad i \in V \text{ and } j = 1, \dots, d. \end{cases} \quad (13)$$

We can finally formalize the control variables[4] $\alpha(t) = (\omega(t), \nu(t), \chi(t))$, that, when paired with the previous system of equations, allows us to view such a system as a neural state model in the form $x'(t) = f(x(t), \alpha(t), t)$, coherently with Eq. (1) and (3). Due to the definition of α, a quadratic penalization in α amounts to a penalization on the "velocities" of the parameters of the network.

Time-Reversed Generalized Riccati Equation

As we briefly discussed in Section , the approach to problem (5) based on Hamilton equations is not usually computationally feasible, mostly due to the fact that it involves boundary conditions on both temporal extrema t_0 and T.

---

[3] We have overloaded the symbol γ: in Eq. (3) was defined as the transition function of the whole state, here only of the y part.

[4] To avoid a cumbersome notation we will denote with the name of a state variable or control variable without specifying any index simply the list of those variables.

More dramatically, Hamilton equations are not generally stable. Consider, for instance, the following example of a widely known control problem:

Example 1 (Linear Quadratic Problem)**.** The Scalar Linear Quadratic problem is obtained by choosing $f(\xi, a, s) = A\xi + Ba$ and $\ell(a, \xi, s) = Q\xi^2/2 + Ra^2/2$ with Q and R positive and $A \in \mathbb{R}$, $B \in \mathbb{R}$. In this specific case, it turns out that the Hamiltonian can be computed in closed form: $H(\xi, \rho, s) = Q\xi^2/2 - B^2\rho^2/(2R) + A\xi\rho$. Hence, the Hamilton equations of Eq. (10) become $x'(t) = -B^2p(t)/R + Ax(t)$ and $p'(t) = -Qx(t) - Ap(t)$. The solution of such system, for general initial conditions, have positive exponential modes $\exp(\omega t)$ with $\omega = \sqrt{A^2 + B^2Q/R}$, that obviously generates instabilities.

However, it turns out that the LQ problem of Example 1 can be approached with a novel solution strategy, which yields stability and is the key element we propose and exploit in this paper to motivate our novel approach to forward-only optimization in neural nets. We can in fact assume that the costate is estimated by $p(t) = \mu(x(t), \theta(t))$, that is defined by $\mu(\xi, \vartheta) = \vartheta\xi$, $\forall(\xi, \vartheta) \in \mathbb{R}^2$. In other words, in this example at each time instant t the costate $p(t)$ is a linear function of the state $x(t)$ with parameter $\theta(t)$. By the definition of the costate, this is equivalent to assume that the value function v is a quadratic function of the state. We then proceed as follows:

1. We randomly initialize $\theta(0)$ and set $x(0) = x^0$;
2. At a generic temporal instant t, under the assumption that $p(t) = \mu(x(t), \theta(t))$, we consider the condition $p'(t) = d\mu(x(t), \theta(t))/dt$ with p' computed with the LQ Hamilton equation (10):

$$\mu_\xi(x(t), \theta(t)) \cdot x'(t) + \mu_\vartheta(x(t), \theta(t)) \cdot \theta'(t) \\ = -Qx(t) - A\theta(t)x(t).$$

Solving this for $\theta'(t)$ we obtain the Riccati equation: $\theta'(t) = (B^2/R)\theta^2(t) - 2S\theta(t) - Q$;

3. We change the sign of the temporal derivative in the Riccati equation

$$\theta'(t) = -(B^2/R)\theta^2(t) + 2S\theta(t) + Q, \quad (14)$$

and we use it with initial conditions to compute $t \mapsto \theta(t)$;

4. Finally, we compute the control parameter using Eq. (11), where the optimal costate is replaced with its estimation given with the network μ.

As it is known, Riccati equation must be solved with terminal conditions; in our case, since we do not have any terminal cost, the optimal solution would be recovered imposing the boundary condition $\theta(T) = 0$. Solving this equation with initial conditions, however, does not have any interpretation in terms of the optimization problem (differently from the forward solution of the costate in Hamilton equations). Instead, let us set for simplicity $t_0 = 0$ and define $\Phi \colon t \in [0,T] \to s \in [0,T]$, with $s := T - t$, so that if we let $\hat{\theta} := \theta \circ \Phi^{-1}$, we have $\forall s \in [0,T]$ that $\hat{\theta}(s) = \theta(\Phi^{-1}(s)) = \theta(T - s)$. This time, $\hat{\theta}$ will satisfy exactly Eq. (14). The solution of this equation with *initial*

condition $\hat{\theta}(0) = 0$ can be found explicitly by standard techniques:

$$\hat{\theta}(s) = \frac{R}{B^2}\lambda_1\lambda_2\frac{e^{\lambda_1 s} - e^{\lambda_2 s}}{\lambda_2 e^{\lambda_1 s} - \lambda_1 e^{\lambda_2 s}},$$

with

$$\lambda_{1,2} = A \pm \sqrt{A^2 + \frac{QB^2}{R}}.$$

This solution has the interesting property that as $T \to \infty$ and $s \to \infty$ with $s < T$ we have that $\hat{\theta}(s) \to \lambda_1 R/B^2$ which is the optimal solution on the infinite temporal horizon. The transformation Φ defined above acts on the temporal domain $[0, T]$ and implements a *reversal of time*, that we can also denote as $t \to T - t$. Given a trajectory on $[0, T]$, applying a time reversal transformation to it (as we did for the parameter θ) entails considering the trajectory in which the direction of time is reversed. The dynamics that we observe while moving forward with this new temporal variable are the same dynamics that we would observe when starting from T and moving backward in the original variable. This comment should also give an intuitive justification of why we could trade final conditions with initial conditions when this transformation is applied.

Neural Costate Estimation

The main contribution of this work is the proposal of a novel method to find a forward approximation of the *costate* trajectory by making use of an additional Feed-forward Neural Network (FNN) to predict its values. We assume that the costate p is estimated by a FNN $\mu(\cdot, \cdot, \vartheta)\colon \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ with parameters $\vartheta \in \mathbb{R}^M$ and then we generalize steps 1–4 that we employed in the LQ problem in the previous subsection as follows: [5]

1. We randomly initialize the parameters of the network μ to the values $\theta(0)$ and select an initial state x^0. This allows us to compute $\mu(x^0, \theta(0))$ and in turn $x'(0)$, using Hamilton equations with $\mu(x^0, \theta(0))$ in place of $p(0)$.[6]

2. At a generic temporal instant t, we assume to know $x(t)$ and $\theta(t)$, we compute $x'(t) = H_\rho(x(t), \mu(x(t), \theta(t)), t)$ and define the loss function (see Remark 2)

$$\Omega_t(\phi) := \frac{1}{2}\|\mu_\xi(x(t), \theta(t)) \cdot x'(t) + \mu_\vartheta(x(t), \theta(t)) \cdot \phi$$
$$+ H_\xi(x(t), \mu(x(t), \theta(t)), t)\|^2 + \frac{\varepsilon}{2}\|\phi\|^2. \tag{15}$$

We choose $\delta\theta(t) \in \arg\min_{\phi \in \mathbb{R}^M} \Omega_t(\phi)$ by performing a gradient descent method on Ω_t.

---

[5]Here we have assumed, mainly to avoid unnecessary long equations, that the $\mu(\cdot, \vartheta)$ take as input only the state; however, more generally its domain could also be enriched with the input signal u. Indeed, in the experimental section we will show some case-studies where this is the case.

[6]Due to the fact that the controls enters in the state equation linearly (see Eq. (13)) if the Lagrangian is quadratic in the controls, like in Eq. (17), then the Hamiltonian (6) can be computed in closed form.

3. We numerically integrate the equation

$$\theta'(t) = -\delta\theta(t) \tag{16}$$

with an explicit Euler step, in order to update the values of θ. We denote this equation (see Remark 4) the *time-reversed generalized Riccati equation*.

4. Finally, we compute the control parameter using Eq. (11) where the optimal costate is replaced with its estimation given with the network μ.

Remark 1. Notice that the assumption that the costate is computed as a function of the state is consistent with its definition in terms of the value function $p(t) = Dv(x(t), t)$. The only real assumption that we are making is that the explicit temporal dependence in $Dv(x(t), t)$ is captured by the dynamic of the parameters $\theta(t)$ of the network μ.

Remark 2. The loss function Ω_t defined in Eq. (15) is designed to enforce the consistency between the following two different estimates of the temporal variations of the costate: *i.* the one that comes from the explicit temporal differentiation of $d\mu(x(t), \theta(t))/dt = \mu_\xi(x(t), \theta(t)) \cdot x'(t) + \mu_\vartheta(x(t), \theta(t)) \cdot \theta'(t)$ and *ii.* the estimate $-H_\xi(x(t), \mu(x(t), \theta(t)), t)$ obtained from the Hamilton equations.

Remark 3. Eq. (16) prescribes a dynamics for the parameters θ that can be interpreted as a time reversal transformation $t \mapsto T - t$ on the dynamics of the parameters of the network μ induced by Hamilton equations (see Remark 2). Our conjecture is that this prescription implements a policy that induces stability to the Hamilton equations (see Section).

Remark 4. Notice that in the LQ case described in the previous section, Eq. (16) indeed reduces to Eq. (14). Indeed, if $\mu(\xi, \vartheta) = \vartheta\xi$ and $\varepsilon \equiv 0$ we have that for LQ $\arg\min_{\phi \in \mathbb{R}^M} \Omega_t(\phi) = \{(B^2/R)\theta^2(t) - 2S\theta(t) - Q\}$, hence $\delta\theta(t) = (B^2/R)\theta^2(t) - 2S\theta(t) - Q$. In this case the equation $\theta'(t) = +\delta\theta(t)$ would be exactly the Riccati equation with the correct sign.

Experiments

In the previous sections, we have presented our proposal within the framework of a continuous time setting. In the subsequent segment of our study, which is dedicated to experimentation, we employ explicit Euler steps of magnitude τ to approximate the differential equations. The number of time steps will be denoted as n_T. Moreover, we assume that the gradient descent procedure mentioned in Sec. is characterized by a number of iterations n_{iter} and a learning rate λ. In appendix B of (Betti et al. 2023) we report a summarizing algorithm of all the procedure presented so far. In order to provide a proof-of-concept of the ideas of this paper, we analyze the capability of our forward-optimization procedure of solving three different tasks with neural estimators: (a) tracking a reference signal, (b) predicting the sign of an input signal and (c) classifying different wave-shapes provided as input signal. The experiences of this section are based on a shared definition of the Lagrangian function ℓ of Eq. (4), which consists of a penalty term on the tracking quality of
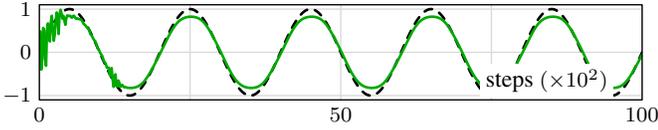
Figure 1: Tracking of a sinusoidal target signal using a recurrent network γ of 2 neurons. Black dashed line: target signal, continuous green line: response of γ. The recurrent network has no input and the target is a sine wave with frequency 0.001 Hz.

Figure 2: Sign prediction of a sinusoidal signal using a recurrent network γ of 2 neurons. Black dashed line: target signal, continuous green line: response of γ. The input of the network is a sinusoidal wave with frequency 0.002 Hz. The target to track is the sign of the input signal.

the target signal, referred to as z, and regularization terms both on the outputs of the neurons in network γ and on the velocities of its parameters, i.e., the control. Formally,

$$\ell(a,\xi,s) = \frac{1}{2}q(\pi(\xi)-z(s))^2 + \frac{1}{2}r_1\sum_{i=p}^{n}\xi_i^2 + \frac{1}{2}r_2\sum_{i=0}^{N}a_i^2, \tag{17}$$

where $z(s)$ is the task-specific target signal at time s and $q,r_1,r_2 \geq 0$ are customizable constant coefficients. We recall that π is a fixed map that, in this case, we assume to simply select one of the neurons in the output of γ (i.e., the first one). Basically, minimizing the Lagrangian implies forcing the output of a neuron, $\pi(\xi(s))$, to reproduce the target signal for every $s \in [t_0,T]$. The goal of our experiences is to find the optimal control α which minimizes the cost functional defined in Eq. (4). In the following subsections we report the results obtained for each experiment, where the initial time step is set to $t_0 = 0$, the outputs of the neurons of γ at the $t_0 = 0$ are initialized to 0, and the parameters of both networks γ and μ start from random values. All the experiments have been conducted using Python 3.9 with PyTorch 2.0.0 on a Windows 10 Pro OS with an Intel Core i7 CPU and 16GB of memory.

Case (a): tracking a target signal Let us consider the case where the target signal is given by $z(s) = \sin(2\pi\varphi s)$, where $\varphi = 0.001$ Hz is the frequency of z, and we want the recurrent network γ to track it. Let us choose the model of the network γ as composed of 2 recurrent neurons fully connected to their inputs, y_0 and y_1, with $\tanh$ activation function, following Eq. (12) (of course, in this experience there is no u). We also downscale the $-y_i$ term by 0.5. Moreover, we choose the network μ as a fully-connected feed-forward net, with 1 hidden layer made up of 20 neurons with ReLU activation functions. The output layer of μ has linear activation. With the choice of $\tau = 0.5$ s, $n_T = 10^4$ time steps, $q = 10^4$, $r_1 = 10^3$, $r_2 = 10^5$, we get the results plotted in Fig. 1. The target signal is the black dashed line, the response of γ is the continuous green line. The number of iterations for updating the derivatives of the weights of μ is set to $n_{\text{iter}} = 100$, with a learning rate $\lambda = 10^{-5}$ and a decay factor $\varepsilon = 10^3$. It is possible to see how the response of γ is able to track the target signal and the accuracy of the tracking quickly improves in the early time steps. The amplitude reached by the response of γ is affected by a slight reduction with respect to z, due to the regularization terms in the Lagrangian function. This experiment confirms that the tracking information,
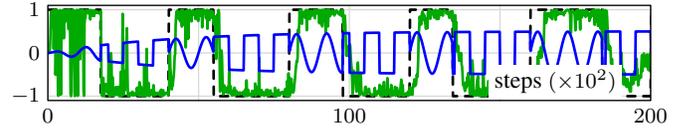
Figure 3: Classification of wave-shapes using a recurrent network γ of 2 neurons. Black dashed line: target signal, continuous green line: response of γ, continuous blue line: input signal. The input of the network is a sequence of sines and square waves, multiplied by a smoothing factor $1 - \exp(-s/\psi)$, where $\psi = 2000\,s^{-1}$.

provided through ℓ, is able to induce appropriate changes in the parameters of γ, that allow such network to follow the signal. Notice that this signal propagates through the state-to-costate map μ and it is not directly attached to the neurons of γ, as in common machine learning problems.

Case (b): predicting the sign of an input signal Let us now assume that both networks γ and μ receive as input a sinusoidal signal $u(s) = \sin(2\pi\varphi s)$ with frequency $\varphi = 0.002$ Hz. The task of predicting the sign of $u(s)$ can be translated in a tracking control problem, where the target signal $z(s)$ is defined as

$$z(s) = \begin{cases} 1, & \text{if } u(s) \geq 0 \\ -1, & \text{otherwise.} \end{cases}$$

Here, we choose the model of the network γ as in Case (a), while the network μ has $\tanh$ activation functions in the hidden layer. With the choice of $\tau = 0.5$ s, $n_T = 1.5 \times 10^4$ time steps, $q = 10^5$, $r_1 = 10^3$, $r_2 = 10^2$, we get the results plotted in Fig. 2. The target signal is the black dashed line, the response of γ is the continuous green line. The maximum number of iterations for updating the derivatives of the weights of μ is again set to $n_{\text{iter}} = 100$, with an adaptive learning rate λ which starts from 10^{-3} and a decay factor $\varepsilon = 10^4$. Here, the adaptive strategy for λ is the one used by the Adam optimizer. This task is clearly more challenging than the previous one, since we ask γ to react in function of the input u, still using the state-to-costate map μ as a bridge to carry the information. Interestingly, also in this case, the response of γ is able to track the target signal, correctly interleaving the information from the previous state and the current input.

Case (c): classifying the different wave-shapes of an input signal Finally, we consider the case in which both
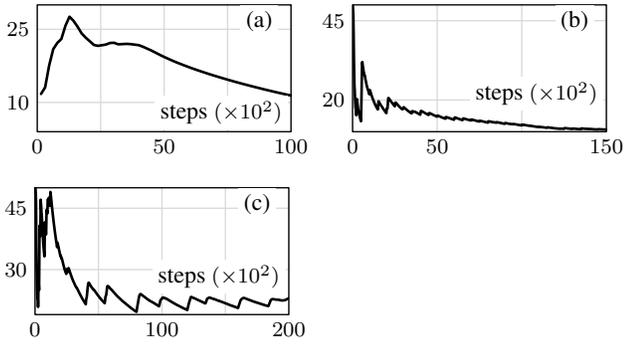
Figure 4: Average value of the Lagrangian function for Case (a), (b) and (c).

Figure 5: Example of lack of generalization of our approach in the wave-shape classification task. Black dashed line: target signal, continuous green line: response of γ, continuous blue line: input signal. The target signal is provided up to 20000 steps and it is masked up to the time horizon.

Figure 6: Example of generalization in the sign prediction task. Black dashed line: target signal, continuous green line: response of γ. The target is given up to 15000 steps.

networks γ and μ get as input a piece-wise defined signal characterized by two different wave-shapes. More precisely, we assume that $u(s) = u_1(s) = (1/2)\sin(2\pi\varphi s)$ or $u(s) = u_2(s) = -(1/2)(-1)^{\lfloor 2\varphi s\rfloor}$, with frequency $\varphi = 0.002$ Hz, in different time intervals randomly sampled on the whole time horizon. Moreover, we multiply $u(s)$ by a smoothing factor $1 - \exp(-s/\psi)$, where $\psi = 2000\,s^{-1}$, in order to help the network μ learning to estimate the costate. The task of classifying the wave-shape of $u(s)$ can be again translated in a tracking control problem, where the target signal $z(s)$ is defined as

$$z(s) = \begin{cases} 1, & \text{if } u(s) = u_1(s) \\ -1, & \text{if } u(s) = u_2(s). \end{cases}$$

In this case, the networks have to deal with the need of differently reacting in different time spans. We choose the models of the networks γ and μ as in Case (b). The maximum number of iterations for updating the derivatives of the weights of μ is again set to $n_{\text{iter}} = 100$, with an adaptive learning rate λ which starts from 10^{-3} and a decay factor $\varepsilon = 10^4$. The adaptive strategy for λ is the same as in Case(b). With the choice of $\tau = 0.5$ s, $n_T = 2 \times 10^4$ time steps, $q = 10^5$, $r_1 = 10^3$, $r_2 = 10^2$, we get the results plotted in Fig. 3. The target signal is the black dashed line, the response of γ is the continuous green line, the input signal is the continuous blue line. Also in this case, the response of γ is able to track the target signal, even if we experienced a small delay in the tracking process that we believe to be motivated by the need of smoothly updating the state, in order to favour the transition in switching from predicting -1 to 1 and vice-versa. In Fig. 4 we report the average value of the Lagrangian for all the tasks that we exposed, obtained dividing the integral of ℓ in $[0, s]$ by s, for each $s \in (0, T]$. It is possible to notice that the mean value of the Lagrangian function decreases as time goes by, reflecting the improvement of the model in tracking the different target signals.

Conclusions and Future Work

This paper introduced a novel theory of optimization that points out a new perspective in the field of optimal control. The forward-in-time Hamiltonian optimization opens up new possibilities for real-time adaptation, tracking and control in lifelong learning scenarios. By bridging the gap between optimal control and deep learning, this innovative methodology paves the way for significant advancements in the learning and adaptation capabilities of autonomous systems in dynamic environments. The paper delved into the theoretical foundations of forward-in-time Hamiltonian optimization, with a particular emphasis on the concept of time-reversed generalized Riccati equation. Future research will focus on enhancing the learning capabilities of the model to facilitate its application in lifelong learning tasks.

In our experiments we have shown that our proposal can be used to efficiently solve different kinds of tracking control problems, where the target signal is always present for each time step. It is important to emphasize that the model is contingent upon a considerable number of parameters and exhibits a high degree of sensitivity to their variations. Consequently, tuning these parameters can be challenging. We recall that gaining explicit generalization capabilities (i.e., when the target signal is not given) is not a goal pursued within the scope of this study, but it will be the main point of our future work. Indeed, we mention that this novel approach still has limitations in such a direction. In most of the experiments, the response of γ is not able to generate the target signal if we mask it after a certain number of time steps, freezing the weights of μ up to the time horizon. An example of this behavior can be seen in Fig. 5. However, in Case (b) we have registered that γ is able to reproduce the target z even if it is masked after 15000 steps, as shown in Fig. 6. This result suggests further investigations on this direction and we will orient our research interest on the capability of learning of our proposal, in order to apply it to lifelong learning tasks (De Lange et al. 2021).

References

Ambrosio, L.; Fusco, N.; and Pallara, D. 2000. *Functions of Bounded Variation and Free continuity Problems*. Oxford Mathematical Monographs, The Clarendon Press Oxford University Press.

Bardi, M.; Dolcetta, I. C.; et al. 1997. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, volume 12. Springer.

Bertsekas, D. 2019. *Reinforcement learning and optimal control*. Athena Scientific.

Betti, A.; Casoni, M.; Gori, M.; Marullo, S.; Melacci, S.; and Tiezzi, M. 2023. Neural Time-Reversed Generalized Riccati Equation. arXiv:2312.09310.

Betti, A.; Faggi, L.; Gori, M.; Tiezzi, M.; Marullo, S.; Meloni, E.; and Melacci, S. 2022. Continual Learning through Hamilton Equations. In *Conference on Lifelong Learning Agents*, 201–212. PMLR.

Courant, R.; and Hilbert, D. 2008. *Methods of mathematical physics: partial differential equations*. John Wiley & Sons.

De Lange, M.; Aljundi, R.; Masana, M.; Parisot, S.; Jia, X.; Leonardis, A.; Slabaugh, G.; and Tuytelaars, T. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7): 3366–3385.

Diehl, M.; Bock, H. G.; and Schlöder, J. P. 2005. A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control. *SIAM Journal on Control and Optimization*, 43(5): 1714–1736.

Evans, L. C. 2022. *Partial differential equations*, volume 19. American Mathematical Society.

Gamkrelidze, R.; Pontrjagin, L. S.; and Boltjanskij, V. G. 1964. *The mathematical theory of optimal processes*. Macmillan Company.

Garcia, C. E.; Prett, D. M.; and Morari, M. 1989. Model predictive control: Theory and practice—A survey. *Automatica*, 25(3): 335–348.

Giaquinta, M.; and Hildebrandt, S. 2013. *Calculus of variations II*, volume 311. Springer Science & Business Media.

Hinton, G. 2022. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*.

Jin, W.; Wang, Z.; Yang, Z.; and Mou, S. 2019. Pontryagin Differentiable Programming: An End-to-End Learning and Control Framework. *ArXiv*, abs/1912.12970.

Lewis, F. L.; Vrabie, D.; and Syrmos, V. L. 2012. *Optimal control*. John Wiley & Sons.

Mai, Z.; Li, R.; Jeong, J.; Quispe, D.; Kim, H.; and Sanner, S. 2022. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469: 28–51.

Osborne, M. R. 1969. On shooting methods for boundary value problems. *Journal of mathematical analysis and applications*, 27(2): 417–433.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Zhang, H.; Wang, Z.; and Liu, D. 2014. A Comprehensive Review of Stability Analysis of Continuous-Time Recurrent Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 25(7): 1229–1262.