Physics-Informed Neural Networks for Closed-Loop Guidance and Control in Aerospace Systems

Conference Paper · January 2022

DOI: 10.2514/6.2022-0361

Physics-Informed Neural Networks for Closed-Loop Guidance and Control in Aerospace Systems

Roberto Furfaro*
Department of System & Industrial Engineering, Department of Aerospace & Mechanical Engineering, University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721

Andrea D'Ambrosio †
School of Aerospace Engineering, Sapienza University of Rome, Via Salaria 851, 00138 Rome, IT
Department of System & Industrial Engineering, University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721

Enrico Schiassi‡, Kristofer Drozd §, and Andrea Scorsoglio ¶
Department of System & Industrial Engineering, University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721

Physics-Informed Neural Networks (PINNs) refer to recently defined a class of machine learning algorithms where the learning process for both regression and classification tasks is constrained to satisfy differential equations derived by the straightforward application of known physical laws. Indeed, Deep Neural Networks have been successfully employed to solve a variety of ODEs and PDEs arising in fluid mechanics, quantum mechanics, just to mention a few. Optimal control problems, i.e. finding a feasible control that minimizes a cost functional while satisfying physical, state and control constraints, are generally difficult to solve and one may need to resort to specialized numerical methods. In this work, we show how PINNs can be employed to synthesize closed-loop optimal guidance and control policies by learning the solution of the Hamilton-Jacobi- Bellman Equation (HJBE) via shallow and deep networks. We show that such methods can be coupled with the Theory of Functional Connections to create numerical frameworks that generate efficient and accurate solutions of the HJBE resulting in novel architectures for closed-loop G&C that may enable autonomy in a large class of aerospace systems.

I. Introduction

The continued and exponential increase in computational capabilities have the potential to enable a new level of autonomy and decision-making for aerospace systems. In particular, the ability to generate real-time, closed-loop trajectories may be considered a technology of utmost importance for management of autonomous aerospace systems within complex environment. Similarly, new analytical and data-driven methods for real-time synthesis of closed-loop optimal controllers may integrate autonomy with robustness and stability. Optimal trajectory generation and closed-loop control rely on optimal control theory which is a well-established discipline. Indeed, given a prescribed model of the dynamical system under consideration, it aims at finding optimal policies and trajectories by minimizing a cost functional that accounts for all constraints and design objectives. Generally, Optimal Control Problems (OCPs) are formulated and subsequently numerically solved using algorithms falling into two broad categories, i.e. direct and indirect methods [1].

Direct methods rely on the transcription of the original OCP into a Non-Linear Programming Problem (NLP). The resulting NLP can be then solved with different numerical optimization algorithms such as the Trust Region Method, Nelder-Mead Method, or Interior Point Methods [2].

Conversely, indirect methods aim at finding the set of differential equations that represent the necessary conditions for the optimal trajectory-control pair. The straightforward application of the Pontryagin Minimum/Maximum Principle

*Department of System & Industrial Engineering, Department of Aerospace and Mechanical Engineering, University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721

†School of Aerospace Engineering, Sapienza University of Rome, Via Salaria 851, 00138 Rome, IT and Department of System & Industrial Engineering, University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721

‡Department of System & Industrial Engineering, University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721

§Department of System & Industrial Engineering, University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721

¶Department of System & Industrial Engineering, University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721

(PMP [3]) or calculus of variation yields a Two-Point Boundary Value Problems in state-costate vectors that must be solved by resorting to available numerical methods. More specifically, PMP is directly applied by building the system control *Hamiltonian* comprising the cost function constrained by the systems dynamics. According to the PMP approach, the optimal control and the first-order optimality necessary conditions are retrieved, and the original OCP is transformed into a TPBVP. The latter is then solved via collocation or shooting methods, together with other numerical methods, such as Runge-Kutta for Ordinary Differential Equations (ODEs) or Finite Elements Methods (FEM) for Partial DEs (PDEs). Although the success of the PMP approach in finding the optimal control and trajectories for a large variety of complex systems, the computed open-loop trajectory is guaranteed to be only locally optimal and its convergence depends upon the selection of a suitable initial guess. Moreover, the computation of open-loop solutions is computationally expensive, making this approach generally infeasible for real-time applications.
Alternatively, necessary and sufficient conditions for optimality can be derived using the *Bellman's Principle of Optimality* which yields the Hamilton–Jacobi–Bellman (HJB) partial differential equation. Indeed, the HJB equation guarantees a necessary and sufficient condition for the optimality while providing a closed-loop solution for the resulting optimal control. The synthesis of the closed-loop controller can be directly obtained once the value function satisfying the PDE is analytically, or most commonly, numerically computed. Importantly, the overall complexity of the HJB equation increases exponentially with the number of dimensions of the system, i.e., the problem is affected by the "curse of dimensionality." Thus, traditional methods to solve PDE, (e.g. by finite element methods), becomes impracticable [4, 5].
Over the past decade, significant advances in machine learning and data analytics have resulted in a dramatic transformation in many scientific fields. Indeed, enabled by the reduction in the cost of sensors, storage and computational resources (e.g. GPUs), a new class of data-driven discovery methods have contributed to transformative and innovative results in characterizing complex non-linear relationships in high-dimensional data collected via experiments.Recently, a new framework named *Physics-Informed Neural Networks* or PINN has been introduced by Raissi et al. [6] which refers to Neural Networks (NN) that are trained by data and constrained to satisfy specific physical laws. Traditional ML methods, including deep NN, are *data-driven*, i.e. they are trained on data to model the unknown non-linear relationship between inputs and outputs. Whereas, PINN-based approaches have shown to have potential in solving open-loop optimal control problems [7, 8], there may be interest in understanding how such methodologies can be adapted to compute closed-loop controllers for aerospace systems.
The goal of this work is to show how PINN-based methodologies can significantly overcome the curse of dimensionality issue for the solution of HJB equations, and therefore make this approach practicable for the closed-loop solution of OCPs in general and aerospace systems in particular. This approach can be used to create a HJB-based controller increasing the autonomy of aerospace GNC systems. More specifically, we will solve HJB equations arising from OCPs with two different Physics-Informed Neural Networks frameworks. The two PINN frameworks that we will be employed to solve the HJB equation are the standard PINN framework, as developed by Raissi et al. [6], and PINN Theory of Functional Connections (PINN-TFC) based methods [9–11]. Furthermore, we will demonstrate that using the PINN-TFC based frameworks, thanks to the properties of the Theory of Functional Connections (TFC), allows computing more accurate solutions than the standard PINN frameworks on a variety of general and aerospace-related closed-loop control synthesis.
The manuscript is organized as follows. First, how to solve OCPs using the HJB approach and PINN frameworks is presented. In the remaining sections, a few selected infinite horizon OCPs are formulated according to the proposed methodology, together with an example describing the application of PINNs to determine closed-loop solutions for integrated guidance and control of exoatmospheric missile intercept.

II. Physics-Informed Neural Networks and Theory of Functional Connections

PINNs are machine learning methods that insert physics into data-driven functional representations of input-output paring collections. As defined by Raissi et al. [6], the term PINN has been coined to indicate those NNs for which the loss function includes the physics as a regularization term. For instance, suppose to face a regression problem using a NN and to have some experimental data representing a physical phenomenon modeled via DEs. The common machine learning approach would be to approximate the data with a NN trained via minimizing the Mean Squared Error (MSE) as the loss function. However, in this case, there are no guarantees that the physics of the problem is not violated. Therefore, PINNs are introduced to take into account the DEs describing the physics behind the collected data. Indeed, the implicit forms of the DEs considered are embedded within the loss function as additional terms, which penalize the training when the DEs and its constraints (e.g., the Initial Conditions or ICs and/or Boundary

Conditions or BCs) are violated. This approach is also referred to as data-physics-driven solution of DEs. In the limit when only the residual of the DEs is considered (e.g., data are not available), PINNs learn the solutions of problems involving Ordinary (ODEs) and Partial (PDEs) DEs in a solely physics-driven fashion. On the other hand, data-driven parameters discovery of DEs (e.g., inverse problems) [6, 12] refers to the estimation of parameters appearing within DEs (e.g., in the orbit determination framework). The major drawback of the standard PINN framework, as introduced by Raissi et al. [6], is that the DE constraints are not analytically satisfied, and therefore, they need to be simultaneously learned with the DE solution within the domain. The PINN-TFC based methods bring a substantial improvement to the standard PINN frameworks. According to these frameworks, the solution of the DEs is approximated using the so-called Constraint Expressions (CEs), which are the key ingredient of the Theory of Functional Connections (TFC), recently developed by Mortari [9]. TFC is a method for functional interpolation where functions are approximated using these CEs. A constrained expression is a functional that is composed by a free-function and a functional that analytically satisfies the constraints regardless of the choice of the free-function [9, 13]. Thanks to this property, TFC has found many applications. TFC is particularly and successfully used for approximating the solution of DEs [14–16]. TFC has already been employed to solve several classes of optimal control space guidance problems via the PMP approach, such as energy optimal landing on large and small planetary bodies [17, 18], and fuel-optimal landing on large planetary bodies [19]. In addition, TFC has been applied to energy optimal trajectories for relative motion problems, such as intercept and rendezvous [20]. The original TFC method, for solving DE, uses, as free-function, a linear combination of orthogonal polynomials. [14, 15]. Using orthogonal polynomials as free-function affects the original TFC framework significantly with the curse of dimensionality when solving large-scale PDEs. To overcome this limitation, PINN-TFC based methods use NNs as free-function. We consider two different architectures for the PINN-TFC based methods. In the first architecture, called Deep-TFC [10], the free function is a deep NN. In the second one, called Extreme-TFC (X-TFC) [11], the free function is a shallow NN trained via the Extreme Learning Machine (ELM) [21] algorithm. However, it is to be noticed that, under the PINN community perspective, according to the definition of Chebyshev NN (ChNN) [22] and Legendre NN (LeNN) [23] the classic TFC applied to the solution of DEs can be classified as X-TFC. Indeed, X-TFC has already been employed for space-related optimal control problems via indirect method and PMP to study constrained and unconstrained optimal intercept problems [8] and planar orbit transfers [7].

III. PINN applied to Optimal Control Problems

In this section we show how to solve OCPs via the solution of the HJB equation using PINN, both in their standard definition [6] and with PINN-TFC based methods. In particular the methodology is applied to the class of infinite horizon optimal control problems with integral quadratic cost.
We consider infinite horizon problems that are governed by linear or non-linear time-invariant affine in the input dynamical systems. Considering a typical LQR integral cost, these problems are posed as following,

$$\min_{\boldsymbol{u}\in\mathcal{U}} J(\boldsymbol{x},\boldsymbol{u}) = \int_0^\infty \left(Q(\boldsymbol{x}) + \boldsymbol{u}^T R\boldsymbol{u}\right) \mathrm{dt} \tag{1}$$

subject to

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}\\ \boldsymbol{x}(0) &= \boldsymbol{x_0}\end{aligned} \tag{2}$$

Where $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^n$, $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^m$ are the states, and control respectively. The system dynamics $\boldsymbol{f} \in \mathbb{R}^n$ and $g(\boldsymbol{x}) \in \mathbb{R}^{n\times m}$ are assumed to be known. The function $Q(\boldsymbol{x})$ is positive definite, and $R \in \mathbb{R}^{n\times m}$ is a symmetric positive definite matrix.
We consider the value function V defined as follows,

$$V(\boldsymbol{x}) = \inf_{\boldsymbol{u}\in\mathcal{U}} J(\boldsymbol{x},\boldsymbol{u}) \tag{3}$$

The value function is the unique solution of the following HJB equation,

$$\sup_{\boldsymbol{u}\in\mathcal{U}} \{-V_{\boldsymbol{x}}^T\dot{\boldsymbol{x}} - \mathcal{L}(t,\boldsymbol{x},\boldsymbol{u})\} = 0 \tag{4}$$

subject to

$$V(\boldsymbol{0}) = 0 \tag{5}$$

where $\mathbf{0}$ is one equilibrium point for the dynamical system. The optimal control $\boldsymbol{u}^*$ is derived as a closed analytical form with respect to the value function V. That is:

$$\boldsymbol{u}^* = \underset{\boldsymbol{u}\in\mathcal{U}}{\operatorname{argsup}}\{-V_{\boldsymbol{x}}^T\dot{\boldsymbol{x}} - \mathcal{L}(t,\boldsymbol{x},\boldsymbol{u})\} = -\frac{1}{2}R^{-1}g^T(x)V_{\boldsymbol{x}} \tag{6}$$

Once $\boldsymbol{u}^*$ is derived, its closed analytical form is plugged back into (4). Then (4) reduces to a nonlinear PDE that is solved for V:

$$\mathcal{R}(\boldsymbol{x},V,V_{\boldsymbol{x}}) = Q(\boldsymbol{x}) + V_{\boldsymbol{x}}^T\boldsymbol{f}(\boldsymbol{x}) - \frac{1}{4}V_{\boldsymbol{x}}^T g(\boldsymbol{x})R^{-1}g^T(x)V_{\boldsymbol{x}} = 0 \tag{7}$$

$$\text{subject to} \quad V(\mathbf{0}) = 0 \tag{8}$$

with $\boldsymbol{x} \in \mathbb{D} \subseteq \mathbb{R}^n$.
The PDE (7) is solved using PINN. The procedure is summarized in figure 1.

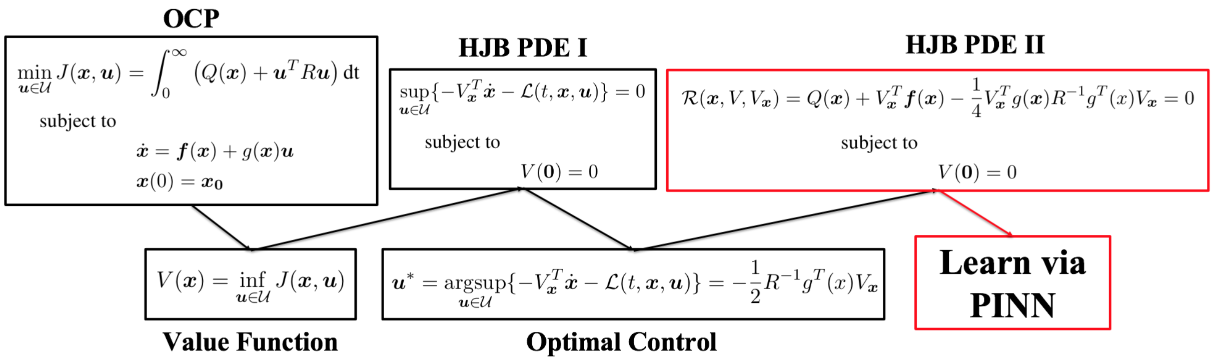

OCP
$\min_{\boldsymbol{u}\in\mathcal{U}} J(\boldsymbol{x},\boldsymbol{u}) = \int_0^\infty (Q(\boldsymbol{x}) + \boldsymbol{u}^T R\boldsymbol{u})\,\mathrm{dt}$
subject to
$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}$
$\boldsymbol{x}(0) = \boldsymbol{x}_0$
HJB PDE I
$\sup_{\boldsymbol{u}\in\mathcal{U}}\{-V_{\boldsymbol{x}}^T\dot{\boldsymbol{x}} - \mathcal{L}(t,\boldsymbol{x},\boldsymbol{u})\} = 0$
subject to
$V(\mathbf{0}) = 0$
HJB PDE II
$\mathcal{R}(\boldsymbol{x},V,V_{\boldsymbol{x}}) = Q(\boldsymbol{x}) + V_{\boldsymbol{x}}^T\boldsymbol{f}(\boldsymbol{x}) - \frac{1}{4}V_{\boldsymbol{x}}^T g(\boldsymbol{x})R^{-1}g^T(x)V_{\boldsymbol{x}} = 0$
subject to
$V(\mathbf{0}) = 0$
$V(\boldsymbol{x}) = \inf_{\boldsymbol{u}\in\mathcal{U}} J(\boldsymbol{x},\boldsymbol{u})$
Value Function
$\boldsymbol{u}^* = \operatorname{argsup}_{\boldsymbol{u}\in\mathcal{U}}\{-V_{\boldsymbol{x}}^T\dot{\boldsymbol{x}} - \mathcal{L}(t,\boldsymbol{x},\boldsymbol{u})\} = -\frac{1}{2}R^{-1}g^T(x)V_{\boldsymbol{x}}$
Optimal Control
Learn via PINN

Fig. 1 Scheme on how to applied PINNs for learning the solution of the HJB equation for infinite horizon OCPs.

According to the standard PINN methodology, V is approximated using a NN,

$$V(t,\boldsymbol{x}) \simeq V_{NN}(t,\boldsymbol{x},\boldsymbol{\theta}) \tag{9}$$

where $\boldsymbol{\theta}$ are the parameters of the NN which are learned via gradient-based methods. To train the network the Mean Square Error (MSE) is to be minimized:

$$\min_{\boldsymbol{\theta}} MSE = MSE_{\mathcal{R}} + MSE_{BC} \tag{10}$$

where,

$$MSE_{\mathcal{R}} = \frac{1}{N_{\mathcal{R}}}\sum_{i=1}^{N_{\mathcal{R}}} \left|\mathcal{R}(\boldsymbol{x}_{\mathcal{R}}^i)\right|^2 \tag{11}$$

$$MSE_{BC} = |V(\mathbf{0}) - V_{NN}(\mathbf{0})|^2 \tag{12}$$

Here $\{\boldsymbol{x}_{\mathcal{R}}^i\}_{i=1}^{N_{\mathcal{R}}}$ represent the collocation points on $\mathcal{R}(\boldsymbol{x})$.
If the nonlinear PDE is solved via a PINN-TFC based method, the constrained expression is,

$$V(\boldsymbol{x}) \simeq V_{\text{CE}}(\boldsymbol{x},\boldsymbol{\theta}) = g(\boldsymbol{x},\boldsymbol{\theta}) + (V(\mathbf{0}) - g(\mathbf{0},\boldsymbol{\theta})) \tag{13}$$

where $\boldsymbol{\theta}$ are the parameters of the NN. It can be clearly seen how the constraint $V(\mathbf{0})$ is analytically satisfied in equation (13). Thus, if PINN-TFC based methods are used, equation (12) is no longer needed in the MSE. This removes one of the main limitation of the standard PINN methods. The fact that the equation constraints are not analytically satisfied

is the cause of having competing objectives during the PINN training. These competing objects are: learning the DE solution within the domain and satisfying the equation constraints. This leads to unbalanced gradients during the network training via gradient-based methods that causes PINNs to have often difficulties to accurately approximate the solution of DE [24]. Indeed, it is well known that gradient-based methods may get stuck in limit cycles or even diverge if multiple competing objectives are present [25, 26]. Another issue of using gradient-based methods is that, as the unknown NN parameters appear non linearly, it is prohibitive to give a good initial guess for nonlinear DEs. Thus, for nonlinear DEs that can be very sensitive to the initial guess on the solution (as many HJB PDEs) the training may diverge, when the DEs are faced solely in a physics-drive fashion.
As previously stated, we use two different architectures for the PINN-TFC based frameworks: Deep-TFC and X-TFC. Deep-TFC uses deep NN as the free-function in the CE. Thus the NN parameters $\boldsymbol{\theta}$ are learned via gradient-based methods. That is, Deep-TFC removes the issue of the competing objects, but it does not remove the issue of the sensitivity to the initial guess for sensitive nonlinear DEs. Conversely, X-TFC uses shallow NN trained with ELM algorithm as free-function in the CE. According to the ELM algorithm, input weights and bias are randomly selected and not tuned during the training, leaving the outputs weight to be the only trainable parameters [21]. Thus, the training is reduced to a faster and more robust least-squares. That is, X-TFC, thanks to the CE and the property of the ELM algorithm, removes both the issues mentioned above. In particular, as the unknowns (e.g., output weights) appear linearly, it possible to provide a good initial guesses when needed.
For the case of OCPs, regardless to the nature of the system dynamics (e.g., linear or nonlinear), the resulting HJB PDE will always be nonlinear. In many cases a good initial guess would be a quadratic value function:

$$V(\boldsymbol{x}) = \boldsymbol{x}^T Q \boldsymbol{x} \tag{14}$$

Learning the solution of the HJB equation arising from the optimal control problem with the aforementioned procedure can enable autonomous optimal control for spacecraft GNC. Indeed, as shown in Figure 2, once the value function is learned and so the optimal control law, the HJB-based controller takes the navigation output as input of the network to generate the control command. This is used by the actuators to provide the required control to the spacecraft. This effectively creates a closed-loop optimal control system. However, one of the limitations of the proposed approach is that the network can only be trained in a finite region of the state space so the controller is accurate only within such region.

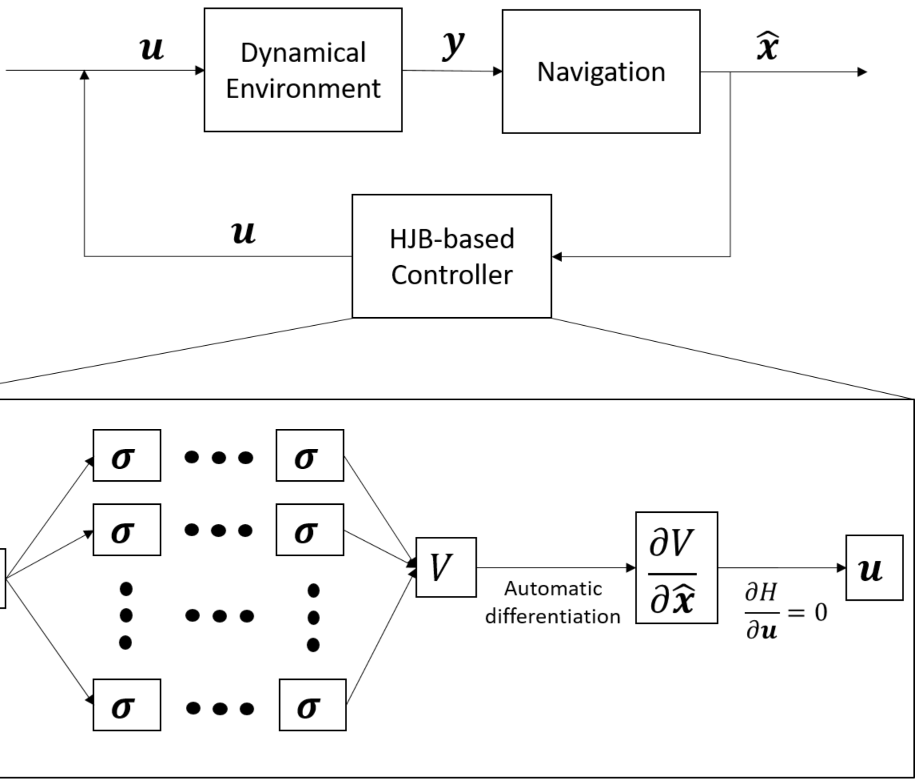


Fig. 2 HJB-based GNC architecture

IV. Selected Optimal Control Problems: Formulation and Results

Many infinite horizon OCPs with integral quadratic cost, both linear and nonlinear, have been solved using the proposed algorithm. Here, for the sake of conciseness we only report three examples. All the problems selected here have analytical solutions. The reason why we selected these problems is to allow the reader to better appreciate the performances and understand the utility of the proposed PINN-based framework; and also to guide the reader through the step-by-step formulation using all the PINN frameworks considered. The formulation and the results are presented and discussed below. The OCPs have coded both in Python 3.7 and ran with an Intel Core i7 - 9700 CPU PC with 64 GB of RAM.

A. Nonlinear, Infinite Horizon Problem 1

Consider the following nonlinear infinite horizon problem (example 2 from [27]),

$$\min \quad \mathcal{J} = \int_0^\infty (\boldsymbol{x}^T \boldsymbol{x} + u^2)\, dt \tag{15}$$

subject to

$$\begin{aligned} \dot{x_1} &= -x_1 + x_2 \\ \dot{x_2} &= -\frac{1}{2}x_1 - \frac{1}{2}x_2 + \frac{1}{2}x_1^2 x_2 + x_1\, u \end{aligned} \tag{16}$$

Using equation (6), the optimal control is,

$$u = -\frac{1}{2}x_1 V_{x_2} \tag{17}$$

Thus the resulting HJB to solve via PINN is,

$$x_1^2 + x_2^2 + (-x_1 + x_2)\, V_{x_1} + \left(-\frac{1}{2}x_1 - \frac{1}{2}x_2 + \frac{1}{2}x_1^2 x_2\right) V_{x_2} - \frac{1}{4}x_1^2 V_{x_2}^2 = 0 \tag{18}$$

subject to the following condition,

$$V(\boldsymbol{0}) = 0 \tag{19}$$

in $x_1 \in [-1, 1]$ and $x_2 \in [-1, 1]$.
The exact solution for the value function is,

$$V_{exact}(\boldsymbol{x}) = \frac{1}{2}x_1^2 + x_2^2 \tag{20}$$

thus the exact optimal control is,

$$u_{\text{exact}}(t, x) = -x_1 x_2 \tag{21}$$

By using the standard PINN framework [6] the value function is approximated as following,

$$V(t, \boldsymbol{x}) \simeq V_{NN}(t, \boldsymbol{x}, \boldsymbol{\theta}) \tag{22}$$

If a PINN-TFC based framework is used, the value function is approximated via the CE. That is,

$$V(t, \boldsymbol{x}) \simeq V_{\text{CE}}(t, \boldsymbol{x}, \boldsymbol{\theta}) = g(t, \boldsymbol{x}, \boldsymbol{\theta}) + (V(\boldsymbol{0}) - g(\boldsymbol{0}, \boldsymbol{\theta})) \tag{23}$$

For this problem $V(\boldsymbol{0}) = 0$, thus the CE is,

$$V(t, \boldsymbol{x}) \simeq V_{\text{CE}}(t, \boldsymbol{x}, \boldsymbol{\theta}) = g(t, \boldsymbol{x}, \boldsymbol{\theta}) - g(\boldsymbol{0}, \boldsymbol{\theta}) \tag{24}$$

where the free function is g is either a DNN trained via gradient based methods, if Deep-TFC is used, or a shallow NN trained via ELM algorithm if X-TFC is used.
We tried different architectures, and the results are summarized in Table 1. For all the architectures $tanh$ was used as activation function. For X-TFC we have also tried ChNN and LeNN. In Table 1 only the absolute training errors are reported, as the absolute test errors are of the same order of magnitude. For all the examples we used 4900 internal training points and one training point at the equilibrium point for standard PINN, while for PINN-TFC based we used

4900 internal training points. In figures 3-4, the value function and the optimal control, computed via X-TFC with 400 neurons, are shown along with the corresponding absolute error with respect to the exact solution. The initial guesses for all the unknown coefficients of X-TFC were set to 0. As can be seen from Table 1, the convergence is pretty good and the results very accurate with respect to the real solutions. In particular, one can note that the epochs in this case correspond to the number of iterations required for the iterative least square procedure. It can be noted that increasing the number of neurons of X-TFC leads to higher training times but does not improve so much the accuracy. Finally, the X-TFC used in with Chebyshev and Legendre polynomial as activation functions (e.g., ChNNs and LeNN, respectively) provides the most accurate results in very low computational times. Indeed, a machine level accuracy is reached in all the four analyzed cases. It is important to note that for ChNNs and LeNNs, the number of neurons corresponds to the degree of the basis function expansion. In figure 5 the loss-epochs history for the standard PINN is reported for a training with 1M epochs for the architecture with the shallow NN and 100 neurons. It can be observed that after a certain number of epochs the training loss shows an asymptotic behaviour. This means that after a certain number of epochs, the learning does not improve. The same asymptotic behaviour was observed for the other architectures tested. It can be observed that between standard PINN and Deep-TFC, the latter provides better results, whereas X-TFC outperforms all the other frameworks, especially when ChNN or LeNN are used.

Table 1 Results for the infinite horizon problem 1: V_{max} is the maximum absolute training error with respect the analytical solution for the value function, $\bar{V}$ is the mean absolute training error with respect the analytical solution for the value function, u_{max} is the maximum absolute training error with respect the analytical solution for the control, $\bar{u}$ is the mean absolute training error with respect the analytical solution for the control

method	layers	neurons	epochs	training time [s]	V_{max}	$\bar{V}$	u_{max}	$\bar{u}$
PINN	1	100	300	5.78	1.06	$1.57 \cdot 10^{-1}$	$5.34 \cdot 10^{-1}$	$1.16 \cdot 10^{-1}$
Deep-TFC	1	100	300	4.29	$5.60 \cdot 10^{-3}$	$1.87 \cdot 10^{-3}$	$1.01 \cdot 10^{-2}$	$1.08 \cdot 10^{-3}$
PINN	1	100	3000	72.14	$5.77 \cdot 10^{-2}$	$2.72 \cdot 10^{-2}$	$8.39 \cdot 10^{-2}$	$1.06 \cdot 10^{-2}$
Deep-TFC	1	100	3000	46.78	$4.76 \cdot 10^{-4}$	$1.95 \cdot 10^{-4}$	$6.93 \cdot 10^{-4}$	$7.16 \cdot 10^{-5}$
PINN	1	100	30000	540.46	$2.21 \cdot 10^{-3}$	$8.35 \cdot 10^{-4}$	$4.21 \cdot 10^{-3}$	$6.65 \cdot 10^{-4}$
Deep-TFC	1	100	30000	431.79	$3.56 \cdot 10^{-5}$	$1.52 \cdot 10^{-5}$	$4.86 \cdot 10^{-5}$	$7.13 \cdot 10^{-5}$
PINN	4	25	30000	647.15	$1.78 \cdot 10^{-3}$	$5.51 \cdot 10^{-4}$	$3.77 \cdot 10^{-3}$	$4.39 \cdot 10^{-4}$
Deep-TFC	4	25	30000	366.99	$9.11 \cdot 10^{-5}$	$2.79 \cdot 10^{-5}$	$1.41 \cdot 10^{-4}$	$2.08 \cdot 10^{-5}$
PINN	10	10	30000	723.09	$1.53 \cdot 10^{-3}$	$6.34 \cdot 10^{-4}$	$2.18 \cdot 10^{-3}$	$4.67 \cdot 10^{-4}$
Deep-TFC	10	10	30000	349.97	$2.15 \cdot 10^{-4}$	$8.41 \cdot 10^{-5}$	$3.47 \cdot 10^{-4}$	$5.78 \cdot 10^{-5}$
PINN	1	200	30000	688.39	$4.72 \cdot 10^{-3}$	$1.22 \cdot 10^{-3}$	$7.79 \cdot 10^{-3}$	$1.13 \cdot 10^{-3}$
Deep-TFC	1	200	30000	687.68	$2.21 \cdot 10^{-5}$	$6.84 \cdot 10^{-6}$	$6.42 \cdot 10^{-5}$	$3.81 \cdot 10^{-6}$
PINN	4	50	30000	823.86	$2.21 \cdot 10^{-3}$	$7.74 \cdot 10^{-4}$	$2.96 \cdot 10^{-3}$	$4.12 \cdot 10^{-4}$
Deep-TFC	4	50	30000	978.15	$6.65 \cdot 10^{-5}$	$2.30 \cdot 10^{-5}$	$8.32 \cdot 10^{-5}$	$1.11 \cdot 10^{-5}$
PINN	10	20	30000	994.40	$2.44 \cdot 10^{-3}$	$6.49 \cdot 10^{-4}$	$4.15 \cdot 10^{-3}$	$5.39 \cdot 10^{-4}$
Deep-TFC	10	20	30000	591.54	$4.40 \cdot 10^{-5}$	$1.87 \cdot 10^{-5}$	$1.11 \cdot 10^{-4}$	$8.21 \cdot 10^{-6}$
PINN	1	100	1000000	10727.54	$1.2 \cdot 10^{-4}$	$2.91 \cdot 10^{-5}$	$2.99 \cdot 10^{-4}$	$4.21 \cdot 10^{-5}$
PINN	4	25	1000000	15001.11	$4.08 \cdot 10^{-4}$	$1.20 \cdot 10^{-4}$	$1.03 \cdot 10^{-3}$	$1.36 \cdot 10^{-4}$
PINN	10	10	1000000	20517.57	$7.23 \cdot 10^{-4}$	$2.22 \cdot 10^{-4}$	$9.76 \cdot 10^{-4}$	$1.26 \cdot 10^{-4}$
X-TFC	1	20	50	6.9375	$6.69 \cdot 10^{-7}$	$2.25 \cdot 10^{-7}$	$2.51 \cdot 10^{-3}$	$2.42 \cdot 10^{-3}$
X-TFC	1	100	50	36.2656	$6.69 \cdot 10^{-7}$	$2.25 \cdot 10^{-7}$	$2.24 \cdot 10^{-6}$	$2.53 \cdot 10^{-7}$
X-TFC	1	400	50	74.6094	$3.23 \cdot 10^{-6}$	$8.79 \cdot 10^{-7}$	$1.40 \cdot 10^{-5}$	$1.22 \cdot 10^{-6}$
X-TFC (ChNN)	1	10	6	1.0313	$2.22 \cdot 10^{-16}$	$2.71 \cdot 10^{-17}$	$3.33 \cdot 10^{-16}$	$1.91 \cdot 10^{-17}$
X-TFC (ChNN)	1	20	6	4.375	$2.22 \cdot 10^{-16}$	$6.01 \cdot 10^{-17}$	$5.55 \cdot 10^{-16}$	$4.90 \cdot 10^{-17}$
X-TFC (LeNN)	1	10	6	1.0781	$4.44 \cdot 10^{-16}$	$4.55 \cdot 10^{-17}$	$4.44 \cdot 10^{-16}$	$2.46 \cdot 10^{-17}$
X-TFC (LeNN)	1	20	6	4.3125	$4.44 \cdot 10^{-16}$	$6.11 \cdot 10^{-17}$	$1.22 \cdot 10^{-15}$	$5.31 \cdot 10^{-17}$

B. Nonlinear, Infinite Horizon Problem 2

Consider the following nonlinear infinite horizon problem (Example 2 from [28])

$$\min \quad \mathcal{J} = \int_0^\infty (\boldsymbol{x}^T \boldsymbol{x} + u^2)\, dt \tag{25}$$

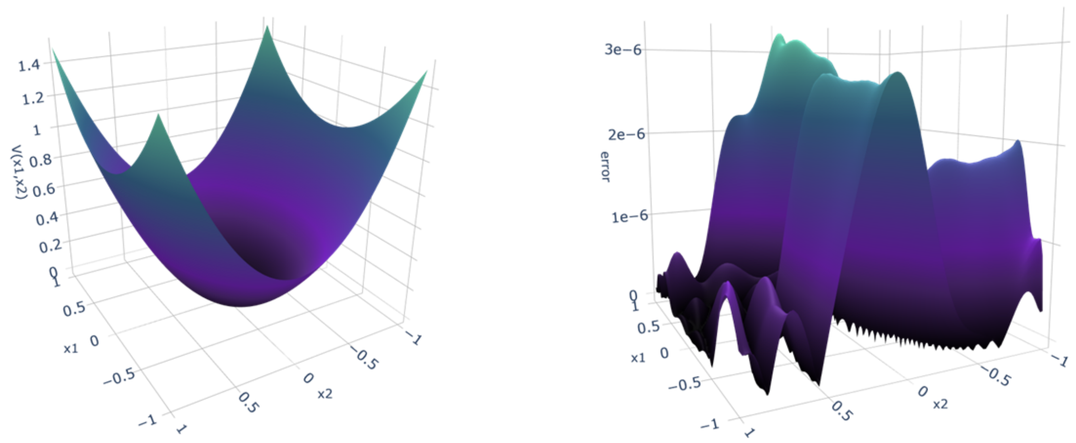


Fig. 3 Value Function for the infinite horizon problem 1 via X-TFC: solution (left) and absolute test error (right).

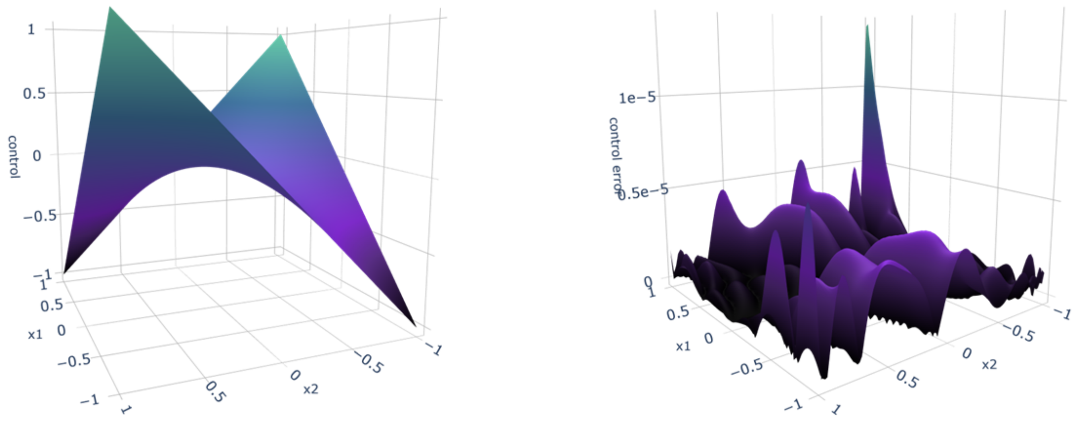


Fig. 4 Optimal Control for the infinite horizon problem 1 via X-TFC: solution (left) and absolute test error (right).

subject to

$$\begin{aligned}\dot{x_1} &= -x_1 + x_2 \\ \dot{x_2} &= -\frac{1}{2}x_1 - \frac{1}{2}x_2\left(1 - (\cos(2x_1)+2)^2\right) + u(\cos(2x_1)+2)\end{aligned} \tag{26}$$

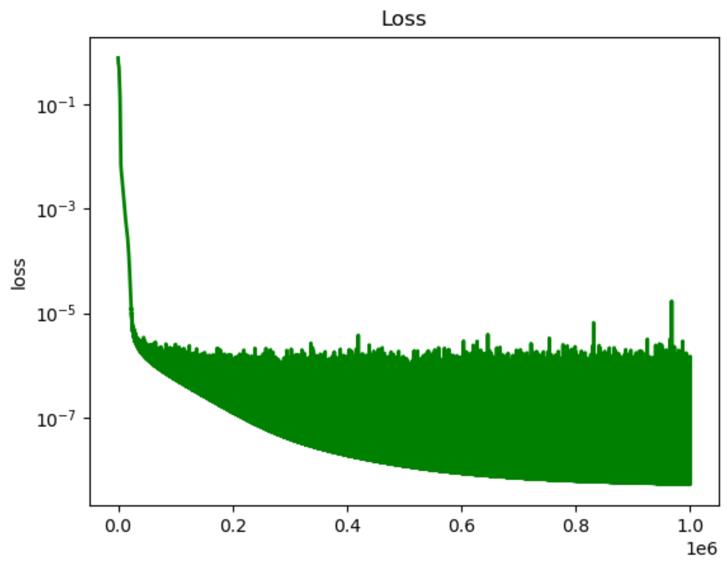


Fig. 5 Infinite horizon problem 1: training Loss for the standard PINN framework for the architecture with shallow NN and 100 neurons.

Using equation (6) again, the optimal control is,

$$u = -\frac{1}{2}(\cos(2x_1) + 2)V_{x_2} \tag{27}$$

Thus the resulting HJB to solve via PINN is,

$$x_1^2 + x_2^2 + (-x_1 + x_2)\, V_{x_1} + V_{x_2}\left(-\frac{1}{2}x_1 - \frac{1}{2}x_2\left(1 - (\cos(2x_1) + 2)^2\right)\right) - \frac{1}{4}V_{x_2}^2\left((\cos(2x_1) + 2)^2\right) = 0 \tag{28}$$

subject to the following condition,

$$V(\mathbf{0}) = 0 \tag{29}$$

in $x_1 \in [-2, 2]$ and $x_2 \in [-2, 2]$.
The exact solution for the value function is,

$$V_{exact}(\boldsymbol{x}) = \frac{1}{2}x_1^2 + x_2^2 \tag{30}$$

thus the exact optimal control is,

$$u_{\text{exact}}(t, x) = -x_2(\cos(2x_1) + 2)^2 \tag{31}$$

The results of this problem are presented in Table 2 and figures 6-7. For this case, both the standard PINN and Deep-TFC failed in learning the solution. This is the reason why only the results obtained via X-TFC are shown. The *tanh* was used as activation function. For X-TFC we have also tried ChNN and LeNN. In Table 2 only the absolute training errors are reported, as the absolute test errors are of the same order of magnitude. For all the examples we used 4900 internal training points. In order to make the training robust enough and less sensitive to choice of the initial guesses, the guesses of the unknown coefficients can be obtained by a simple regression of the following generic quadratic function: $V_{guess} = (x_1 + x_2)^2$, as already shown in Eq. (14). In figures 6-7, the value function and the optimal control, computed via X-TFC with 800 neurons, are shown along with the corresponding absolute error with respect to the exact solution. Even for this case, ChNN and LeNN provide machine level accuracy errors with respect to the true solution. For the generic X-TFC, it is possible to notice that increasing the number of neurons just increases the computational time of the training but does not lead to a better accuracy.

Table 2 Results for the infinite horizon problem 2: V_{max} is the maximum absolute training error with respect the analytical solution for the value function, $\bar{V}$ is the mean absolute training error with respect the analytical solution for the value function, u_{max} is the maximum absolute training error with respect the analytical solution for the control, $\bar{u}$ is the mean absolute training error with respect the analytical solution for the control

method	layers	neurons	epochs	training time [s]	V_{max}	$\bar{V}$	u_{max}	$\bar{u}$
X-TFC	1	20	50	5.9843	$5.17 \cdot 10^{-3}$	$1.46 \cdot 10^{-3}$	$4.43 \cdot 10^{-3}$	$9.23 \cdot 10^{-4}$
X-TFC	1	100	50	31.2188	$1.68 \cdot 10^{-4}$	$4.22 \cdot 10^{-5}$	$2.29 \cdot 10^{-4}$	$4.74 \cdot 10^{-5}$
X-TFC	1	800	50	171.0	$1.25 \cdot 10^{-4}$	$5.96 \cdot 10^{-5}$	$2.31 \cdot 10^{-4}$	$4.47 \cdot 10^{-5}$
X-TFC (ChNN)	1	10	6	0.9375	$8.88 \cdot 10^{-16}$	$1.70 \cdot 10^{-16}$	$8.88 \cdot 10^{-16}$	$7.34 \cdot 10^{-17}$
X-TFC (ChNN)	1	20	6	4.0938	$1.77 \cdot 10^{-15}$	$3.06 \cdot 10^{-16}$	$2.66 \cdot 10^{-15}$	$3.03 \cdot 10^{-16}$
X-TFC (LeNN)	1	10	6	0.8438	$1.78 \cdot 10^{-15}$	$2.76 \cdot 10^{-16}$	$1.78 \cdot 10^{-15}$	$1.07 \cdot 10^{-16}$
X-TFC (LeNN)	1	20	6	4.3125	$2.66 \cdot 10^{-15}$	$6.02 \cdot 10^{-16}$	$4.44 \cdot 10^{-15}$	$3.38 \cdot 10^{-16}$

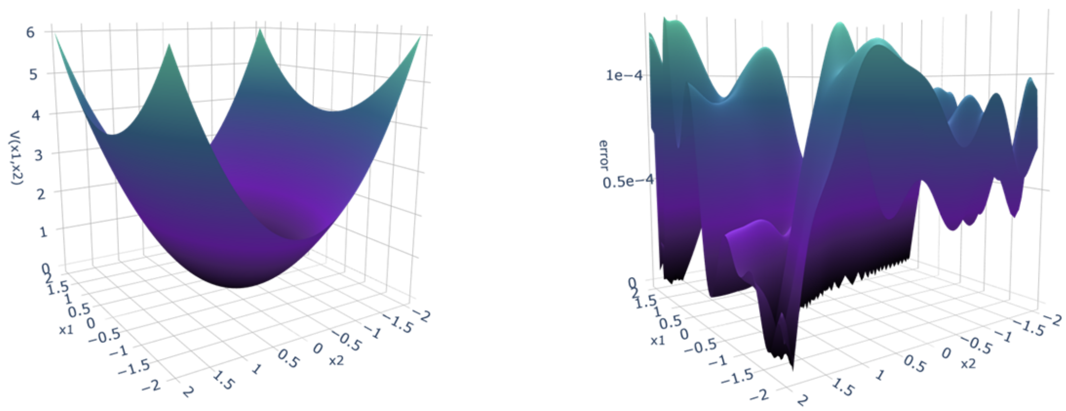
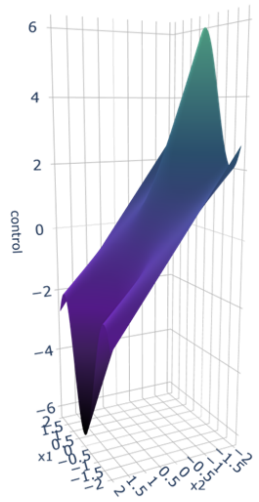


Fig. 6 Value Function for the infinite horizon problem 2 via X-TFC: solution (left) and absolute test error (right).



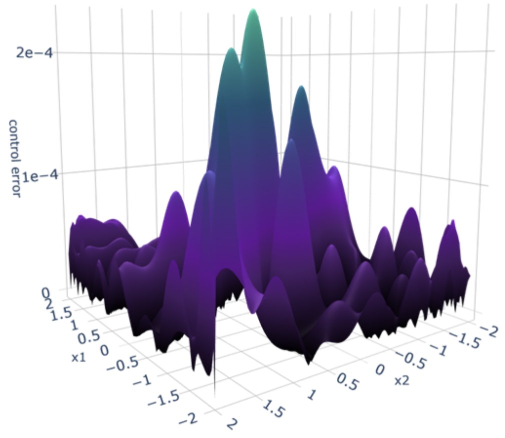


Fig. 7 Optimal Control for the infinite horizon problem 2 via X-TFC: solution (left) and absolute test error (right).

C. Infinite Horizon Optimal Control in Newton Mechanics

The third example is taken from [29]. It is a classic example of OCP in Newton mechanics.

$$\min \quad \mathcal{J} = \frac{1}{2}\int_0^\infty (\boldsymbol{x}^T\boldsymbol{x} + u^2)\, dt \tag{32}$$

subject to

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= u \end{aligned} \tag{33}$$

where x_1, x_2, and u represent the particle's position, velocity, and acceleration respectively. Using equation (6), the optimal control is,

$$u = -V_{x_2} \tag{34}$$

Thus the resulting HJB to solve via PINN is,

$$-\frac{1}{2}x_1^2 + -\frac{1}{2}x_2^2 - V_{x_1}x_2 + \frac{1}{2}V_{x_2}^2 = 0 \tag{35}$$

subject to the following condition,

$$V(\mathbf{0}) = 0 \tag{36}$$

in $x_1 \in [-1, 1]$ and $x_2 \in [-1, 1]$.
The exact solution for the value function is,

$$V_{exact}(\boldsymbol{x}) = \frac{\sqrt{3}}{2}x_1^2 + \frac{\sqrt{3}}{2}x_2^2 + x_1x_2 \tag{37}$$

thus the exact optimal control is,

$$u_{\text{exact}}(t, x) = -\sqrt{3}x_2 - x_1 \tag{38}$$

The results of this problem are presented in Table 3 and figures 8-9. For this case, both the standard PINN and Deep-TFC failed in learning the solution. This is the reason why only the results obtained via X-TFC are shown. The $tanh$ was used as activation function. For X-TFC we have also tried ChNN and LeNN. In Table 3 only the absolute training errors are reported, as the absolute test errors are of the same order of magnitude. For all the examples we used 4900 internal training points. In order to make the training robust enough and less sensitive to choice of the initial guesses, the guesses of the unknown coefficients can be obtained by a simple regression of the following generic quadratic function: $V_{guess} = \frac{1}{2}(x_1 + x_2)^2$, as already shown in Eq. (14). In figures 8-9, the value function and the optimal control, computed via X-TFC with 400 neurons, are shown along with the corresponding absolute error with respect to the exact solution. Even for this case, ChNN and LeNN provide machine level accuracy errors with respect to the true solution. It is worthy to notice that better solutions are obtained with less neurons in the case of ChNN and LeNN. This is probably caused by an over-fitting. For the generic X-TFC, it is possible to notice that increasing the number of neurons just increases the computational time of the training but does not lead to a better accuracy.

Table 3 Results for the infinite horizon problem in Newton Mechanics: V_{max} is the maximum absolute training error with respect the analytical solution for the value function, $\bar{V}$ is the mean absolute training error with respect the analytical solution for the value function, u_{max} is the maximum absolute training error with respect the analytical solution for the control, $\bar{u}$ is the mean absolute training error with respect the analytical solution for the control

method	layers	neurons	epochs	training time [s]	V_{max}	$\bar{V}$	u_{max}	$\bar{u}$
X-TFC	1	20	50	4.625	$2.76 \cdot 10^{-3}$	$4.28 \cdot 10^{-4}$	$3.96 \cdot 10^{-3}$	$6.33 \cdot 10^{-4}$
X-TFC	1	100	50	23.2656	$1.02 \cdot 10^{-6}$	$8.69 \cdot 10^{-9}$	$4.76 \cdot 10^{-6}$	$3.98 \cdot 10^{-8}$
X-TFC	1	400	50	68.875	$1.44 \cdot 10^{-7}$	$6.42 \cdot 10^{-10}$	$8.51 \cdot 10^{-7}$	$3.66 \cdot 10^{-9}$
X-TFC (ChNN)	1	10	6	0.7344	$2.66 \cdot 10^{-15}$	$1.25 \cdot 10^{-16}$	$1.15 \cdot 10^{-14}$	$2.02 \cdot 10^{-16}$
X-TFC (ChNN)	1	20	5	3.2031	$2.10 \cdot 10^{-12}$	$4.81 \cdot 10^{-15}$	$1.94 \cdot 10^{-11}$	$4.83 \cdot 10^{-14}$
X-TFC (LeNN)	1	10	5	0.7344	$7.55 \cdot 10^{-15}$	$1.34 \cdot 10^{-16}$	$3.69 \cdot 10^{-14}$	$3.61 \cdot 10^{-16}$
X-TFC (LeNN)	1	20	5	3.375	$2.33 \cdot 10^{-12}$	$4.93 \cdot 10^{-15}$	$2.29 \cdot 10^{-11}$	$5.31 \cdot 10^{-14}$

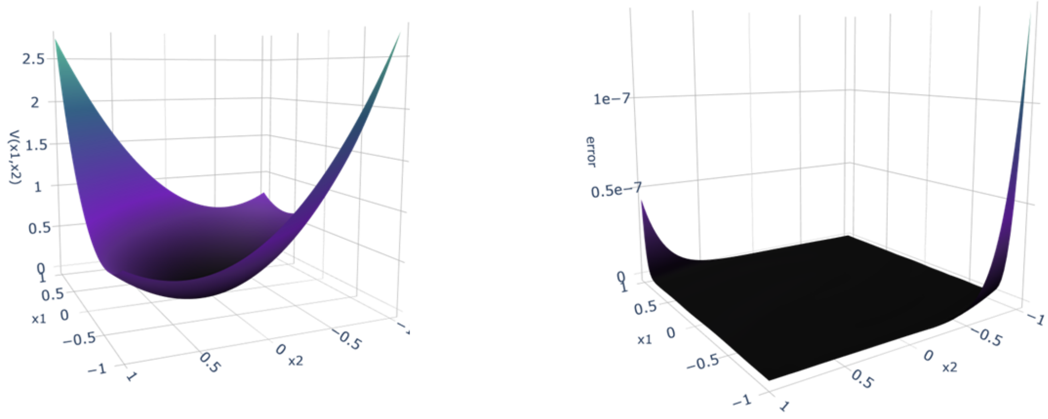


Fig. 8 Value Function for the infinite horizon problem in Newton Mechanics via X-TFC: solution (left) and absolute test error (right).

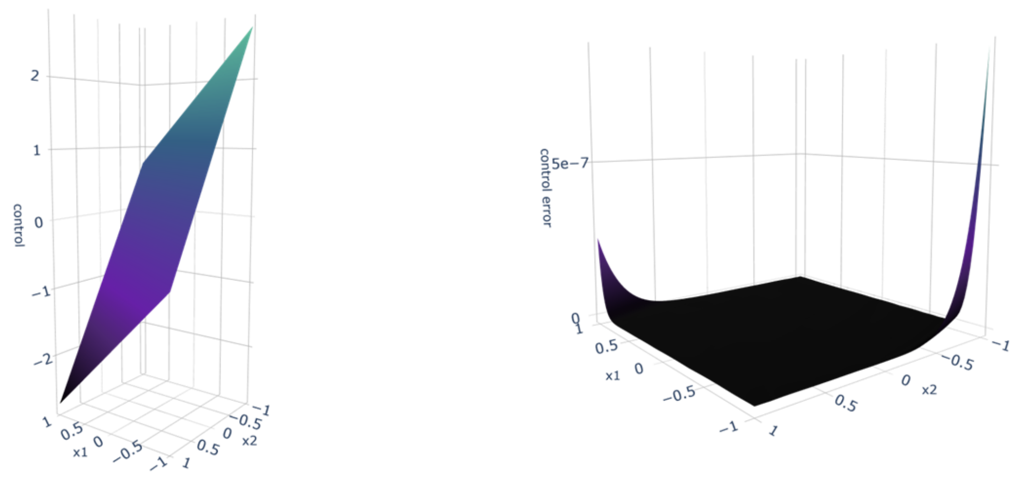


Fig. 9 Optimal Control for the infinite horizon problem in Newton Mechanics via X-TFC: solution (left) and absolute test error (right).

D. Discussions

The results prove the feasibility of using PINN frameworks for solving HJB PDEs for the class of infinite horizon OCPs considered in this work. The main advantage in using NN based methods to solve this kind of problems is that once the optimal control is learned, with sufficient accuracy, on the training points, thanks to the approximation properties of the NNs, there is no need to compute it again if we want to evaluate it on points unseen during the training (e.g., test points). Thus, we have a closed-loop solution for the OCP that can be used to compute optimal trajectories in real-time. Another significant advantage is that PINN methods are not affected by the curse of dimensionality when solving large-scale PDEs [11]. For this reason, PINN methods are very appealing for solving HJB PDEs for OCPs. Indeed for many OCPs of interest, especially for space applications, the arising HJB is a large-scale PDEs (e.g., the dimension of the PDE is greater than two).
In this research, we employed two different PINN frameworks: the classic (or standard) one as developed by Raissi et al. [6], and PINN-TFC based frameworks. The difference is that with PINN-TFC based methods, the unknown solution is approximated with the so-called CE defined within the original TFC. Using the CE, the equations are analytically satisfied, and therefore they do not need to be added as an additional term in the MSE. We tried two different architectures for the PINN-TFC based frameworks: Deep-TFC and X-TFC. Among all the PINN frameworks tested (e.g., standard PINN, Deep-TFC, and X-TFC), X-TFC seems to be the best suitable and robust, at least for the class of OCPs considered in the paper. This is because the X-TFC does not require gradient-based methods for the NN training, allowing for a good initial guess when required. Within X-TFC, we tested several activation functions: classic activation functions such as hyperbolic tangent, ChNN, and LeNN. The results show that ChNN and LeNN outperform the classic activation functions by several magnitudes in terms of accuracy. However, ChNN and LeNN become prohibitive for large-scale problems (e.g., with more than two independent variables). Thus, X-TFC with classic activation functions appears to be the best trade-off for solving HJB PDEs in this class of problems.

V. Closed-loop solution of Linear Quadratic Problem via Physics Informed Neural Networks: Applications to Integrated G&C for Missile Intercept

PINN methods can be shown to be effective in solving general optimal linear problems with quadratic cost functions. In this section, we apply X-TFC to fastly and accurately solve the matrix differential Riccati equation arising from such problems. More specifically, we demonstrate the methodology by applying it to the problem of designing an integrated guidance and flight control system of interceptor missiles. Traditional missile architectures are designed assuming that spectral separation between guidance and control holds, which may not be guaranteed especially close to interception due to rapid geometric changes during the end-game phase. Integrated G&C assume that one single loop is needed. Here, the guidance law relates directly to the dynamics of the airframe instantiating the full state feedback of the missile internal states. Consequently, the guidance law is computed as a solution of a finite-time control problem which can be solved directly using a PINN-based methodology, yielding a closed-loop solution approximated via a NN that satisfy the physics of the missile engagement.

A. Linear Quadratic Regulator

For linear-quadratic OCPs, rather than solving the HJB PDE, another way in which a closed-loop solution can be obtained for is by solving the Matrix Differential Riccati Equation (MRDE). The solution of the MRDE for an optimal controller is the main component of the Linear Quadratic Regulator (LQR) feedback controller. A finite-horizon, continuous time linear quadratic problem assumes the following cost function

$$\mathcal{J} = \boldsymbol{x}^T(t_f)\boldsymbol{P}_f\boldsymbol{x}(t_f) + \int_{t_0}^{t_f} \boldsymbol{x}^T(t)\boldsymbol{Q}x(t) + \boldsymbol{u}^T(t)\boldsymbol{R}\boldsymbol{u}(t)\mathrm{d}t, \tag{39}$$

where $\boldsymbol{Q} \geq 0$, $\boldsymbol{R} > 0$, $\boldsymbol{P}_f \geq 0$ are symmetric, positive (semi-) definite matrices chosen upon some desirable criteria. For example, $\boldsymbol{Q}$ and $\boldsymbol{R}$ weight the importance of the state and control being driven toward 0 throughout the trajectory. Furthermore, the $\boldsymbol{P}_f$ matrix weights the importance of only the terminal state being driven toward 0. The LQR problem involves minimizing Eq. (39) by the correct selection of $\boldsymbol{u}$. Here, the system dynamics is governed by linear differential equations of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}. \tag{40}$$

One could solve this OCP via Pontryagin's Minimum Principle:

$$\begin{aligned} \mathcal{H} &= \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u} + \lambda^T(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) \\ 0 &= \frac{\partial\mathcal{H}}{\partial\boldsymbol{u}} = \boldsymbol{R}\boldsymbol{u} + \lambda^T\boldsymbol{B} \to \boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\lambda. \end{aligned} \tag{41}$$

The necessary conditions of optimally are then satisfied by solving the following two-point boundary value problem (TPBVP),

$$\dot{\boldsymbol{x}} = \frac{\partial\mathcal{H}}{\partial\lambda} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}; \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{42a}$$

$$\dot{\lambda} = -\frac{\partial\mathcal{H}}{\partial\boldsymbol{x}} = -\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}^T\lambda; \quad \lambda(t_f) = \boldsymbol{P}_f\boldsymbol{x}(t_f). \tag{42b}$$

Note that the solution of the above TPBVP is also sufficiently optimal because of the Strengthened Legendre-Clebsch Condition (i.e., $\mathcal{H}_{\boldsymbol{uu}} > 0$). Instead of solving the TPBVP, one can guess the form of the solution to be

$$\lambda(t) = \boldsymbol{P}(t)\boldsymbol{x}(t), \tag{43}$$

which when plugged into Eq. (42b) gives

$$\begin{aligned} \dot{\lambda} &= \dot{\boldsymbol{P}}\boldsymbol{x} + \boldsymbol{P}\dot{\boldsymbol{x}} \\ &= \dot{\boldsymbol{P}}\boldsymbol{x} + \boldsymbol{P}\left(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}\right)\boldsymbol{x} \\ &= \dot{\boldsymbol{P}}\boldsymbol{x} + \boldsymbol{P}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{x} = -\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{P}\boldsymbol{x}. \end{aligned} \tag{44}$$

Simplifying the above equation yields the MRDE,

$$-\dot{\boldsymbol{P}} = \boldsymbol{P}\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{Q}; \quad \boldsymbol{P}(t_f) = \boldsymbol{P}_f. \tag{45}$$

The MRDE is an ordinary differential equation with a terminal constraint that can be solved backwards in time. When solved, the closed-loop solution of the OCP can then be determined,

$$\boldsymbol{u}(t) = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}(t)\boldsymbol{x}. \tag{46}$$

B. Matrix Riccati Differential Equation Solution via a Physics Informed Neural Network

LQR controllers have been around for quite some time. Therefore, researchers have developed numerous methods to solve the MRDE. However, the most commonly employed and simple method for solving the MRDE is direct integration. For example, one can select any of the Runge-Kutta (RK) schemes to perform the integration. One of the main disadvantages of directly integrating the MRDE with RK schemes is that for suitable accuracy it requires a small step size and may yields numerically unstable solutions. Here, we use the X-TFC approach and we approximate the components of the P matrix by a shallow network using chebyshev polynomials (ChNN). The residual of the MRDE is minimized via iterative least-square until convergence. Generally, given the selected finite-time interval where the solution in sought, one needs to define the numbers of collocation (training) points within the global constrained expressions. Sensitivity of the solution to the number of training points yields two possible approaches, i.e. 1) systematically increasing the number of collocation points or 2) successively solve many Final Value Problems (FVP), (i.e. domain decomposition). Our initial attempt at 1) showed that the accuracy of the X-TFC solution was not sufficiently adequate thus diverting our attention to the domain decomposition approach. Essentially, we split the problem domain into many equidistant sub intervals. The first sub interval was from t_f to some t_n. The MRDE over this sub interval could then be solved and the next MRDE would solved from t_n to t_{n-1}, where the final condition $\boldsymbol{P}(t_n)$ was found via the previous (FVP). All FVPs were then solved successively until the solution of the last FVP over $[t_0, t_1]$ was found. Combining the solution of each FVP then gave the solution to the MRDE over the entire problem domain. Figure 10 provides a graphical interpretation of the successive FVP approach.

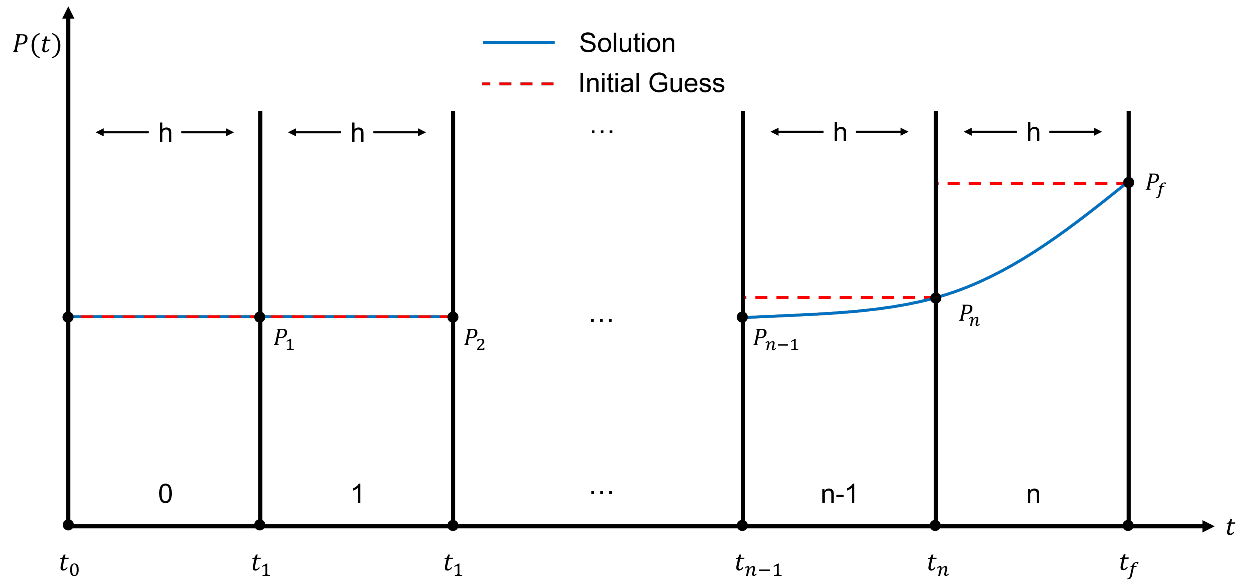


Fig. 10 Successive FVP Diagram.

C. Application to Integrated G&C for Exoatmospheric Missile Intercept

The test case we use to demonstrate XTFC's ability to approximate the MDRE solution is taken from Ref. [30]. More specifically, the scenario involves the end-game stage of the missile intercept problem where the actor implements thrust vector control (TVC). Integrating the MDRE, even with X-TFC, to formulate a guidance law is referred to as an integrated single-loop guidance law. In other words, the guidance law (i.e., Eq. 46) is related directly to the dynamics of the airframe, as in Figure 11 and a full state feedback on the missile's internal states is employed.

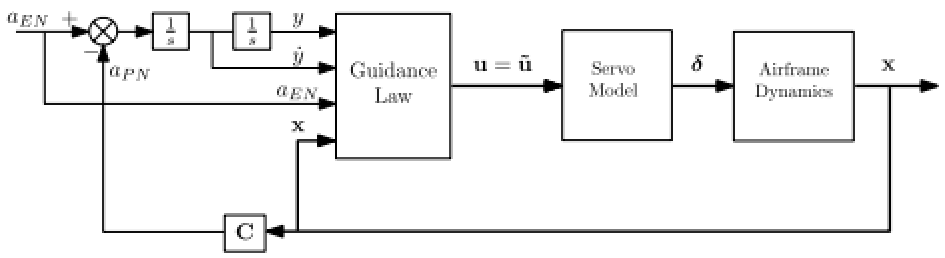


Fig. 11 Integrated single-loop guidance law diagram from Ref. [30].

The end-game TVC missile test case is planar as is shown in Figure 12. Furthermore, the missile orientation is given in Figure 13 The state is given by

$$\boldsymbol{x} = \begin{Bmatrix} y & \dot{y} & a_{\text{EN}} & \theta & \dot{\theta} & \delta_t \end{Bmatrix}^T, \quad \tilde{u} = u = \delta_t^c,$$

where y is the displacement between the target and the missile normal to the X axis, a_{EN} is the evader acceleration normal to LOS$_0$, θ represents the missile's body orientation, δ_t is the missile's thrust deflection, and δ_t^c is the commanded thrust deflection. The weight matrices are

$$\boldsymbol{Q} = \boldsymbol{0}, \quad \boldsymbol{R} = 1, \quad \boldsymbol{P}_f = \begin{bmatrix} 1 \times 10^{-8} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}.$$

Lastly the state-space form of the equation set is given by

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{A}_{22} \end{bmatrix}, \quad \boldsymbol{A}_{11} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_{12} = \begin{bmatrix} 0 & 0 & 0 \\ -120 & 0 & -120 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_{22} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -200 \\ 0 & 0 & -10 \end{bmatrix}, \quad \&$$

$$\boldsymbol{B} = \begin{Bmatrix} \boldsymbol{0} & 10 \end{Bmatrix}^T.$$

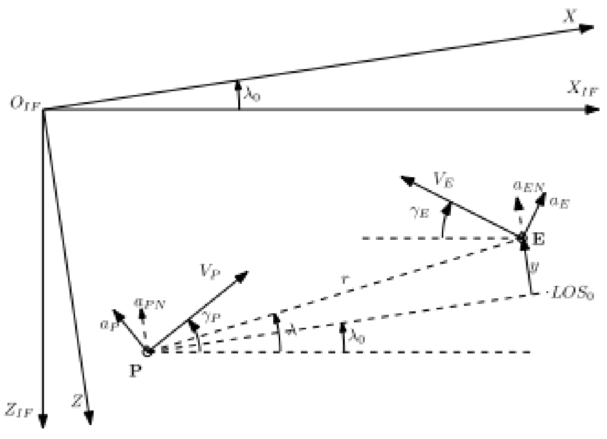


Fig. 12 Planar engagement geometry from Ref. [30].

D. Results

X-TFC was successfully implemented to approximate the MDRE solution. A time horizon of $h = 0.001$ seconds was selected to solve each FVP successively. We used 20 neurons and 30 training points for each FVP and a ChNN

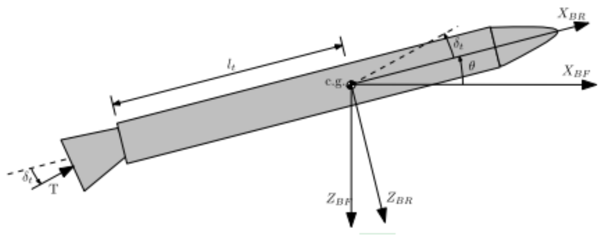


Fig. 13 Missile orientation from Ref. [30].

was selected as the free function. Statistics that demonstrate X-TFC's performance at solving the MDRE are given in Table 4. Each FVP was solved in less than 4 iterations on average and the computation speed of each FVP was on an order of milliseconds. Furthermore, the L_2 norm of each FVP solution was on the order of $O(10^{-16})$. Figure 14 shows the trajectories of diagonal elements pertaining to the MDRE solution found via X-TFC. Test points are shown on the plot in order to demonstrate that the computed constrained expressions used to approximate the solutions are proficient at interpolation. Lastly, Figures 15 and 16 give the state and control trajectories found by applying Eq. (46). A comparison with the results of Ref. [30] shows that they are equivalent.

Table 4 Results for the end-game TVC missile test case: $\bar{\mathcal{L}}$ is the average L_2 norm of the XTFC MDRE solutions and $\tilde{\mathcal{L}}$ is the corresponding standard deviation.

method	layers	neurons	# training points	# of FVPs	mean iterations	mean training time [s]	$\lvert\bar{\mathcal{L}}\rvert_2$	$\lvert\tilde{\mathcal{L}}\rvert_2$
X-TFC (ChNN)	1	20	30	3000	3.27	0.0623	$2.1158 \cdot 10^{-16}$	$1.4346 \cdot 10^{-16}$

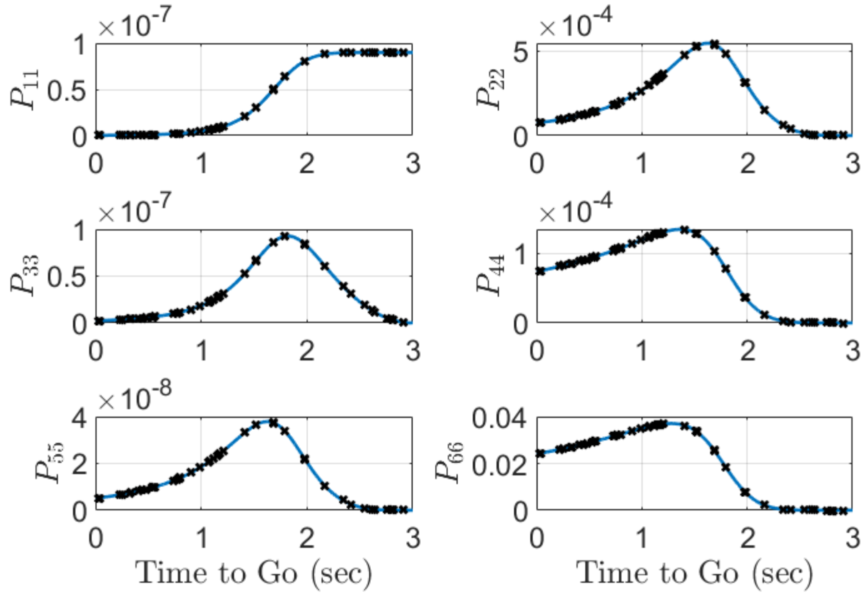


Fig. 14 Diagonal elements of the approximated MDRE solution that was computed with XTFC. The x marks indicate test points.

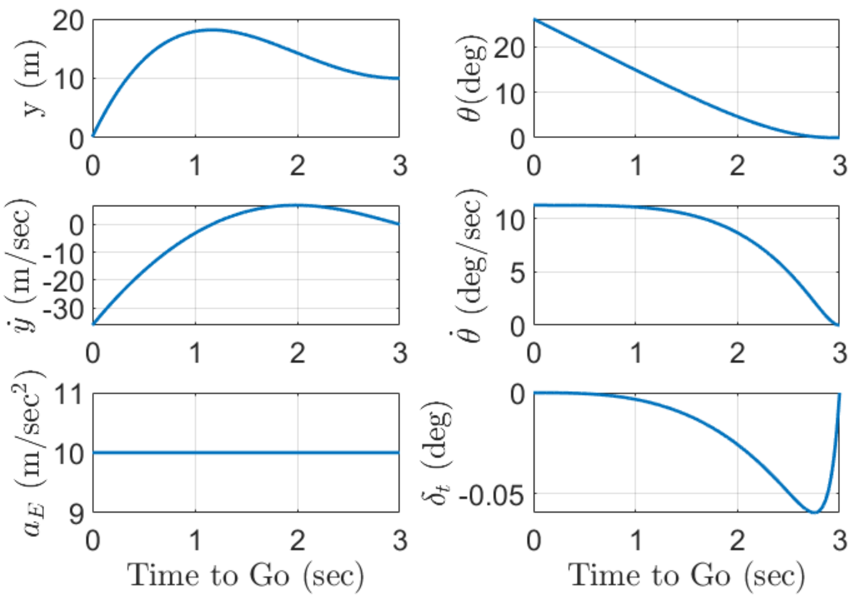


Fig. 15 State trajectories from the XTFC MDRE Solution.

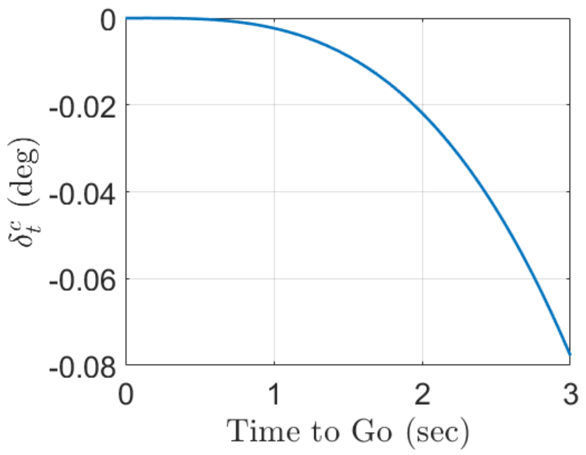


Fig. 16 Control trajectory from the XTFC MDRE Solution.

VI. Conclusions

The next generation of autonomous aerospace systems will require novel architectures and techniques that integrate data-driven and physics-based methods. Here, we successfully demonstrated that PINN approaches could be employed to solve HJB PDE for infinite horizon OCPs and be exploited for future aerospace applications. More specifically, we considered the class of infinite horizon OCPs with integral quadratic cost. There are two main advantages in using PINN based methods for solving these kinds of problems: once the optimal control is learned on the training points, it can be used to compute optimal trajectories in real-time, even on points unseen during the training without the need to be learned again, and large scale HJB PDEs (e.g., PDEs with more than two independent variables) can be tackled as PINN methods are not affected by the curse of dimensionality.
We considered two different PINN approaches in this work, i.e., the classic one, as defined by Raissi et al., and the PINN-TFC based methods. Based on our results, PINN-TFC based methods (particularly X-TFC) seem preferable to the classic PINN to solve this class of OCPs.

Concerning the solution of HJB PDE, we solved three benchmark infinite horizon OCPs to prove the feasibility of

PINN-based methods in solving these kinds of problems. However, efforts are in progress to solve large-scale OCPs belonging to the class considered here, especially for space applications. On the other hand, we have also shown that the PINN-TFC framework can successfully be used to generate a closed-loop control by learning the solution of the Matrix Differential Riccati Equation. To improve the accuracy of the results, the domain decomposition technique has been exploited. The overall approach has been tested with the aerospace application regarding the end-game stage of the missile intercept problem, obtaining good results.

Future works will focus on extending these PINN-based frameworks to solve the HJB PDE related to other aerospace applications and to tackle different class of OCPs, such as problems with discontinuous control (e.g., bang-bang type control).

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Rao, A. V., "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, Vol. 135, No. 1, 2009, pp. 497–528.

[2] Poe, W. A., and Mokhatab, S., *Modeling, control, and optimization of natural gas processing plants*, gulf professional publishing, 2016.

[3] Ross, I. M., *A primer on Pontryagin's principle in optimal control*, Collegiate Publ., 2009.

[4] Chilan, C. M., and Conway, B. A., "Optimal nonlinear control using Hamilton–Jacobi–Bellman viscosity solutions on unstructured grids," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 1, 2020, pp. 30–38.

[5] Cristiani, E., and Martinon, P., "Initialization of the shooting method via the Hamilton-Jacobi-Bellman approach," *Journal of Optimization Theory and Applications*, Vol. 146, No. 2, 2010, pp. 321–346.

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

[7] Schiassi, E., D'Ambrosio, A., Johnston, H., De Florio, M., Drozd, K., Furfaro, R., Curti, F., and Mortari, D., "Physics-Informed Extreme Theory of Functional Connections Applied to Optimal Orbit Transfer," *Proceedings of the Astrodynamics Specialist Conference, Lake Tahoe, CA, USA*, 2020, pp. 9–13.

[8] D'Ambrosio, A., Schiassi, E., Curti, F., and Furfaro, R., "Pontryagin Neural Networks with Functional Interpolation for Optimal Intercept Problems," *Mathematics*, Vol. 9, No. 9, 2021, p. 996.

[9] Mortari, D., "The Theory of Connections: Connecting Points," *MDPI Mathematics*, Vol. 5, No. 57, 2017.

[10] Leake, C., and Mortari, D., "Deep theory of functional connections: A new method for estimating the solutions of partial differential equations," *Machine learning and knowledge extraction*, Vol. 2, No. 1, 2020, pp. 37–55.

[11] Schiassi, E., Leake, C., De Florio, M., Johnston, H., Furfaro, R., and Mortari, D., "Extreme Theory of Functional Connections: A Physics-Informed Neural Network Method for Solving Parametric Differential Equations," *arXiv preprint arXiv:2005.10632*, 2020.

[12] Schiassi, E., D'Ambrosio, A., De Florio, M., Furfaro, R., and Curti, F., "Physics-Informed Extreme Theory of Functional Connections Applied to Data-Driven Parameters Discovery of Epidemiological Compartmental Models," *arXiv preprint arXiv:2008.05554*, 2020.

[13] Mortari, D., and Leake, C., "The Multivariate Theory of Connections," *MDPI Mathematics*, Vol. 7, No. 3, 2019, p. 296. https://doi.org/https://doi.org/10.3390/math7030296.

[14] Mortari, D., "Least-squares Solution of Linear Differential Equations," *MDPI Mathematics*, Vol. 5, No. 48, 2017, pp. 1–18. https://doi.org/10.3390/math5040048, URL http://www.mdpi.com/2227-7390/5/4/48.

[15] Mortari, D., Johnston, H., and Smith, L., "High accuracy least-squares solutions of nonlinear differential equations," *Journal of Computational and Applied Mathematics*, Vol. 352, 2019, pp. 293 – 307. https://doi.org/https://doi.org/10.1016/j.cam.2018.12.007, URL http://www.sciencedirect.com/science/article/pii/S0377042718307325.

[16] Leake, C., Johnston, H., and Mortari, D., "The Multivariate Theory of Functional Connections: Theory, Proofs, and Application in Partial Differential Equations," *Mathematics*, Vol. 8, No. 8, 2020, p. 1303.

[17] Furfaro, R., and Mortari, D., "Least-squares solution of a class of optimal space guidance problems via Theory of Connections," *Acta Astronautica*, 2019. https://doi.org/https://doi.org/10.1016/j.actaastro.2019.05.050, URL http://www.sciencedirect.com/science/article/pii/S0094576519302292.

[18] Schiassi, E., D'Ambrosio, A., Johnston, H., Furfaro, R., Curti, F., and Mortari, D., "COMPLETE ENERGY OPTIMAL LANDING ON SMALL AND LARGE PLANETARY BODIES VIA THEORY OF FUNCTIONAL CONNECTIONS," *2020 AAS/AIAA Astrodynamics Specialist Conference*, 2020.

[19] Johnston, H., Schiassi, E., Furfaro, R., and Mortari, D., "Fuel-Efficient Powered Descent Guidance on Large Planetary Bodies via Theory of Functional Connections," *The Journal of the Astronautical Sciences*, under review.

[20] Drozd, K., Furfaro, R., Schiassi, E., Johnston, H., and Mortari, D., "Energy-optimal trajectory problems in relative motion solved via Theory of Functional Connections," *Acta Astronautica*, Vol. 182, 2021, pp. 361–382.

[21] Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K., " Extreme learning machine: Theory and applications ," *Neurocomputing*, Vol. 70, No. 2006, 2006, pp. 489–501. https://doi.org/10.1016/j.neucom.2005.12.126.

[22] Liu, M., Hou, M., Wang, J., and Cheng, Y., "Solving two-dimensional linear partial differential equations based on Chebyshev neural network with extreme learning machine algorithm," *Engineering Computations*, 2020.

[23] Yang, Y., Hou, M., and Luo, J., "A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods," *Advances in Difference Equations*, Vol. 2018, No. 1, 2018, p. 469.

[24] Wang, D. Y., *Study Guidance and Control for Lunar Soft Landing (Ph.D. Dissertation)*, School of Astronautics, Harbin Institute of Technology, Harbin, China, 2000.

[25] Mertikopoulos, P., Papadimitriou, C., and Piliouras, G., "Cycles in adversarial regularized learning," *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2018, pp. 2703–2717.

[26] Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., and Graepel, T., "The mechanics of n-player differentiable games," *International Conference on Machine Learning*, PMLR, 2018, pp. 354–363.

[27] Tang, W., and Daoutidis, P., "Distributed adaptive dynamic programming for data-driven optimal control," *Systems & Control Letters*, Vol. 120, 2018, pp. 36–43.

[28] Vamvoudakis, K. G., and Lewis, F. L., "Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, Vol. 46, No. 5, 2010, pp. 878–888.

[29] Cacace, S., Cristiani, E., Falcone, M., and Picarelli, A., "A patchy dynamic programming scheme for a class of Hamilton–Jacobi–Bellman equations," *SIAM Journal on Scientific Computing*, Vol. 34, No. 5, 2012, pp. A2625–A2649.

[30] Levy, M., Shima, T., and Gutman, S., "Linear quadratic integrated versus separated autopilot-guidance design," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 6, 2013, pp. 1722–1730.