DOI: 10.13009/EUCASS2023-056
Aerospace Europe Conference 2023 – 10TH EUCASS – 9TH CEAS

Embedding Pontryagin's Principle in Neural Networks for Optimal Asteroid Landing

Sergio Cuevas del Valle[⋆†]*, Pablo Solano-López*[⋆] *and Hodei Urrutxua*[⋆]
[⋆]*Aerospace Systems and Transport Research Group, Universidad Rey Juan Carlos*
Camino del Molino 5, 28942, Fuenlabrada, Spain
s.cuevas.2017@alumnos.urjc.es · pablo.solano@urjc.es · hodei.urrtxua@urjc.es
[†]Corresponding author

Abstract

This work proposes novel Neural Network (NN) training algorithms and architectures to solve with low-cost general Optimal Control problems: regression of the optimal control policy for a given state trajectory. This is achieved via two novelties: first, a novel cost-effective optimal control solver is used for low-cost data augmentation of optimal state-control trajectories, and then combined with Feedforward and General Regression Neural Networks to solve the so-called direct regression problem. Secondly, Pontryagin's Maximum Principle is leveraged to modify physics-informed NN to embed the Hamiltonian structure of Optimal Control within the training algorithm for enhanced robustness and generalized performance.

1. Introduction

The advent of Machine Learning (ML) and its applications within engineering in general and to space engineering in particular, has allowed to rethink complex problems in space dynamics, ranging from learning dynamical models from telemetry to spacecraft operations and decision taking in general.[1]

It is more and more common to find such techniques as a standard methodology towards the design of general guidance, navigation, and control systems.[2] Among others, ML techniques have been applied as solvers of complex optimal control problems, such as orbital rendezvous and transfers,[3] stationkeeping in complex environments,[4] or landing[5,6] and trajectory planning under general uncertainty. Particularly, Neural Networks (NN) models, thanks to their universal approximation capabilities,[7] can be constructed to learn these optimal solutions and then generalize them to new problems of similar statement.

Optimal control problems are natural and intrinsically valuable to astrodynamics and general mission design, in which tight performance constraints exist. Despite the vast literature on the topic, Optimal Control Theory still an active field of research, especially with regards to real time performance onboard legacy systems, a main focus of interest for the past years.[8] The design of these optimal control laws is constrained by the need to solve complex nonlinear programming (NLP) problems associated to a Hamiltonian Minimization Condition (HMC), either in the form of Pontryagin's Maximum Principle or the complementary Hamilton-Jacobi-Bellman PDE equation.[9] This is true for both the so-called direct and indirect approaches,[10] in which discretization is leveraged to solve the primal or dual optimal problem, respectively. The direct approach is easier to solve but suffers from the curse of dimensionality; the indirect method provides a rigorous way to verify optimality but is subject to intrinsic instability arising from its symplectic structure (which is however exploited in our work). In both cases computational capabilities are required to solve the associated nonlinear programming optimization, which may not always be available. In this sense, NNs offer an alternative to find the same solutions, constructed upon memory storage and not online performance, and have recently been gaining attention from the community as general, low-cost solvers (once the training has been accomplished).[11] Moreover, their use as natural embeddings of actual abstract optimality conditions has recently become a topic of high interest,[12,13] precisely the focus of this paper.

Based on recent advances in the field,[14,15] this work proposes novel Neural Network training algorithms and architectures to solve general Optimal Control problems at minimum computational cost. In particular, a high-efficiency Optimal Control solver[16] is used as a novel dataset generator and data augmentation technique, providing massive learning data for supervised algorithms at null expenses when compared to traditional optimization software. The combination of this solver with General Regression Architectures is shown to provide enhanced performance when compared to classical deep learning networks in scarce data scenarios. Moreover, the Hamiltonian structure of Optimal Control (given by the application of Pontryagin's Maximum Principle) is leveraged to modify physics-informed NN[17]

DOI: 10.13009/EUCASS2023-056

to the control practitioner's advantage. This HMC embedding is used to construct and train the network in a robust and generalizable manner. These training algorithms are designed to output the optimal control law associated to a given cost function and state point. In this way, with low computational effort once the network is trained, the major difficulty in the Optimal Control indirect approach is removed, immediately making available the optimal state-control tuple while naturally addressing the required optimality conditions.

The remainder of this paper is organized as follows. Section 2 introduces an abstract formulation of the most general optimal problem, whose solution process is the main focus of our work. Once introduced, two particularisations of this Bolza problem are presented in Section 3, namely, the Linear Quadratic Regulator (LQR) and the problem of optimal landing and descent on an ellipsoidal asteroid. While the novelties presented aimed to be general, the scope of this investigation is restricted to a provide a Proof of Concept of the conceptual design of the optimal control solver proposed. Moreover, Pontryagin's Maximum Principle and the theoretical Optimal Control corpus needed to introduce such solver is briefly revisited in Section 4. The generation of the training datasets and the needed methodology to do so is covered in Section 5, to finally present the proposed deep learning optimal control architecture in Section 6. Finally, verification and validation of the network is achieved in Section 7, while steps ahead and open lines of research discussed in Section 8.

2. Optimal Control and the General Bolza Problem

The focus of this paper is to propose a novel solving technique for realizations of the following general optimal control Bolza problem

$$\begin{aligned} \underset{\mathbf{s},\,\mathbf{u}}{\arg\min} \quad & J = G\left(\mathbf{s}(t_f),\, \mathbf{s}(0),\, t_f,\, t_0\right) + \int_{t_0}^{t_f} l\left(\mathbf{s}, \mathbf{u}, t\right) \mathrm{d}t \\ \text{subject to} \quad & \dot{\mathbf{s}} = \mathbf{f}(\boldsymbol{\mu}, \mathbf{s}, \mathbf{u}) \,, \\ & \mathbf{s}(t_0) = \mathbf{s}_0 \,, \\ & \mathbf{e}\left(\mathbf{s}(t_f), t_f\right) = \mathbf{0} \,, \\ & \mathbf{g}(\boldsymbol{\mu}, \mathbf{s}) = \mathbf{0} \,, \\ & \mathbf{h}(\boldsymbol{\mu}, \mathbf{s}) < \mathbf{0} \,, \end{aligned} \tag{1}$$

where the state of the dynamical system is described by the vector $\mathbf{s}$ and whose first-order evolution with respect to the independent variable t is governed by the vector field $\mathbf{f}$, characterized by a set of parameters $\boldsymbol{\mu}$ and the control vector field $\mathbf{u}$.

The solution is given by the determination of the phase space flow $\mathbf{s}^*(t)$ and control application $\mathbf{u}^*(t)$ minimizing the cost function J while satisfying the boundary conditions equalities $\mathbf{g}$ and path constraints $\mathbf{h}$.

As it has already been introduced, solving such general problem is not usually an easy task, and advanced computational machinery is needed in the general case to solve an equivalent general NLP, which is the result of discretizing the problem into a finite set of algebraic constraints. This work tries to address a systematic procedure to obtain and solve for the necessary optimality conditions of the problem and embed them into a neural network architecture, thus transforming computational into simply memory requirements.

To our advantage in the developments to follow, the Mayer term $G\left(\mathbf{s}(t_f),\, \mathbf{s}(0),\, t_f,\, t_0\right)$ in the cost function J will be absorbed by the Lagrange, running cost function l, so that the previous general Bolza problem is transformed into that of Lagrange

$$\begin{aligned} \underset{\mathbf{s},\,\mathbf{u}}{\arg\min} \quad & J = \int_{t_0}^{t_f} l\left(\mathbf{s}, \mathbf{u}, t\right) \mathrm{d}t \\ \text{subject to} \quad & \dot{\mathbf{s}} = \mathbf{f}(\boldsymbol{\mu}, \mathbf{s}, \mathbf{u}) \,, \\ & \mathbf{s}(t_0) = \mathbf{s}_0 \,, \\ & \mathbf{e}\left(\mathbf{s}(t_f), t_f\right) = \mathbf{0} \,, \\ & \mathbf{g}(\boldsymbol{\mu}, \mathbf{s}, \mathbf{u}) = \mathbf{0} \,, \\ & \mathbf{h}(\boldsymbol{\mu}, \mathbf{s}, \mathbf{u}) < \mathbf{0} \,. \end{aligned} \tag{2}$$

The rationale behind such re-casting is that the Mayer term cannot be estimated online easily in real-time scenarios, while the Lagrange penalty term, given its integral nature, is actually trivially computed while executing the control law $\mathbf{u}$. However, the terminal conditions associated to G (vital in Dynamic Programming, for example[18]) are lost in the process.

DOI: 10.13009/EUCASS2023-056

3. Toy models examples: linear quadratic regulation and optimal asteroid landing

While the final objective of this work is to provide a general solver neural network architecture, in this first communication the scope of the analysis is restricted to two main dynamical frameworks, which are of prime interest on their own: linear quadratic regulation and optimal asteroid landing.

3.1 Linear Quadratic Regulation

Arguably, Optimal Control Theory was born and mainly developed as a byproduct of technological needs of the Space Race, and therefore Aerospace applications have remained as one of its major focus since its infancy times. This specially applies to space dynamics optimization, in which optimality plays a fundamental role in mission design and operations.

Quadratic optimization (those problems in which the cost function J is a quadratic form) are the workhorse in space optimization. While they may not be always representative of the underlying problem of interest, specially when used as a proxy to fuel consumption,[19] their major advantage is the availability of close-form, analytical solutions, among which the Kalman's Linear Quadratic Regulator (LQR) stands out.[9] Naturally, the LQR will provide our first validation and verification problem.

In particular, the linear LQR problem is defined by

$$\begin{aligned} \underset{\mathbf{s},\mathbf{u}}{\arg\min} \quad & J = \frac{1}{2}\int_{t_0}^{t_f} \mathbf{s}^\intercal Q\,\mathbf{s} + \mathbf{u}^\intercal R\,\mathbf{u}\,\mathrm{d}t \\ \text{subject to} \quad & \dot{\mathbf{s}} = A\,\mathbf{s} + B\,\mathbf{u}\,, \\ & \mathbf{s}(t_0) = \mathbf{s}_0\,, \\ & \mathbf{s}(t_f) = \mathbf{0}\,. \end{aligned} \tag{3}$$

The linear model (A, B) will be detailed in Section 5. In-depth solutioning of the linear LQR may be found in almost all primer texts to Optimal Control Theory and is here ommited for brevity. In short, the optimal control application is given by the following feedback law

$$\mathbf{u} = -R^{-1}B^\intercal S\,\boldsymbol{s} = -K\mathbf{s}\,.$$

The matrix S is the positive solution to the following Ricatti differential equation

$$\dot{S} + Q + SA + A^\intercal S - SBR^{-1}B^\intercal S = 0\,, \quad S(t_f) = 0\,.$$

3.2 Optimal asteroid landing

Great part of the motivation of this paper is previous work by Solano-López and Barea on optimal asteroid landing.[14] In their work, neural networks were used to provide first initial estimates of the time-optimal costate initial conditions, from which the complete landing state trajectory and control policy may be obtained.

The dynamics of the problem, which feature a modified Newtonian gravity field, are given by

$$\ddot{\mathbf{r}} = \mathbf{g} + \Gamma\mathbf{u}\,,$$

where the function $\mathbf{g}$ characterizes the asteroid gravity field, Γ is the thrust or instantaneous control magnitude and $\mathbf{u}$ denotes the thrusting vector.

The fundamental reference frame in which the position vector $\mathbf{r}$ is realised is chosen with its origin at the centre of mass of the body and its axes aligned with the ellipsoid principal axes of inertia.

The gravity field $\mathbf{g}$ is described by

$$\mathbf{g} = \kappa\,\mathbf{r}\int_{\xi}^{\infty}\frac{\mathrm{d}\alpha}{\left(a_i^2+\alpha\right)\phi}\,, \quad \phi = \sqrt{\left(a_x^2+\alpha\right)\left(a_y^2+\alpha\right)\left(a_z^2+\alpha\right)}\,.$$

The parameter κ characterizes the intensity of the gravity attraction, playing the role of Gauss' constant in Newtonian gravity. The variable ϕ is further introduced to simplify notation. Finally, ξ represents the positive confocal ellipsoidal coordinate associated to $\mathbf{r}$, defined as the single positive root of the following equation:

$$\sum_i^3 \frac{r_i^2}{a_i^2+\xi^2} = 1\,.$$

DOI: 10.13009/EUCASS2023-056

The final optimal asteroid landing problem is given by

$$\begin{aligned} \underset{\mathbf{s},\Gamma,\mathbf{u}}{\arg\min} \quad & J = \int_{t_0}^{t_f} \mathrm{d}t \\ \text{subject to} \quad & \ddot{\mathbf{r}} = \mathbf{g} + \Gamma\mathbf{u}\,, \\ & \mathbf{s}(t_0) = \mathbf{s}_0\,, \\ & \mathbf{s}(t_f) = \left[\mathbf{r}_f^{\intercal},\, \mathbf{0}\right]^{\intercal}\,, \\ & \Gamma < \Gamma_{\max}\,. \end{aligned} \tag{4}$$

The inclusion of the maximum thrust magnitude constraint ensures that the control set $\mathbb{U}$ is a subset of the Euclidean space $\mathbb{R}^3$; otherwise, for time-optimal problems, an unconstrained control authority leads to unbounded optimal control laws: infinite thrusting indeed results in null time of flight.

4. Pontryagin's Maximum Principle

Pontryagin's Maximum Principle (PMP) is the systematic, analytical procedure leading to the optimality necessary (but not sufficient) conditions of any general Bolza problem.[9] Despite its procedural nature, PMP comes with associated difficulties (mainly, solving a sensitive Two Boundary Value Problem), and therefore still requires of state of the art computational techniques to be appropriately solved. In fact, modern optimal control solvers do not solve PMP (named problem B^{λ}), but only uses it as an optimality test of the solutions obtained via a direct transcription of the problem (or B^{N} in the technical argot).[8]

As we have already introduced, the focus of this communication is to establish a neural network architecture in which PMP is naturally encoded, which results in two main advantages: a) elimination of the need to analytically establish the particularisation of the PMP to the problem of interest, which is not always a trivial exercise; b) elimination of the need of further computational resources to solve the outcome necessary conditions.

PMP is fundamentally constructed upon the definition of an appropriate Hamiltonian to the problem of Bolza, and proposes associated instantaneous minimization conditions, which are equivalent to the necessary optimality ones. Such Hamiltonian is given by

$$H = l\left(\mathbf{s}, \mathbf{u}, t\right) + \boldsymbol{\lambda}^{\intercal}\mathbf{f}\left(\boldsymbol{\mu}, \mathbf{s}, \mathbf{u}, t\right)\,. \tag{5}$$

The co-states $\boldsymbol{\lambda}$ map the dynamics vector field to the integral running cost function and, despite the common opinion, they have physical significance and dimensionality (with units of cost over time). The co-states are actually covectors or linear forms, which provide a valid metric (dot product) in the tangent space of the phase space of the problem, in which $\mathbf{f}$ lives.

The core of the PMP set of necessary conditions is the following Hamiltonian Minimization Conditions (HMC)

$$\begin{aligned} \underset{\mathbf{u}}{\arg\min} \quad & H\left(\boldsymbol{\lambda}, \mathbf{s}, \mathbf{u}, t\right) \\ \text{subject to} \quad & \mathbf{u} \in \mathbb{U}\,, \end{aligned} \tag{6}$$

where $\mathbb{U}$ is the problem's control set. Moreover, its symbolic representation is commonly leveraged

$$\mathbf{u}^{*} = \underset{\mathbf{u}\in\mathbb{U}}{\arg\min}\, H(\boldsymbol{\lambda}, \mathbf{s}, \mathbf{u}, t)$$

The stationary point of H with respect to the pair $(\mathbf{s}, \boldsymbol{\lambda})$ naturally encodes the dynamics of the problem via an adjoint system of equations

$$\begin{aligned} \frac{\partial H}{\partial \boldsymbol{\lambda}} = \mathbf{0} \rightarrow \dot{\mathbf{s}} = \mathbf{f}\left(\boldsymbol{\mu}, \mathbf{s}, \mathbf{u}, t\right)\,, \\ \frac{\partial H}{\partial \mathbf{s}} = -\dot{\boldsymbol{\lambda}}\,. \end{aligned} \tag{7}$$

The final constraints given by PMP are the terminal or transversality conditions, together with the Hamiltonian

DOI: 10.13009/EUCASS2023-056

Value and Evolution Conditions.

$$\mathbf{s}(0) = \mathbf{s}_0\,, \tag{8}$$

$$\mathbf{e}\left(\mathbf{s}(t_f), t_f\right) = \mathbf{0}\,, \tag{9}$$

$$\boldsymbol{\lambda}(t_f) = \frac{\partial E}{\partial \mathbf{s}_f}\,, \tag{10}$$

$$H^*\left(\boldsymbol{\lambda}_f, \mathbf{s}_f, \mathbf{u}_f, t_f\right) = -\frac{\partial E}{\partial t_f}\,, \tag{11}$$

$$\frac{\mathrm{d}H^*}{\mathrm{d}t} = \frac{\partial H}{\partial t}\,. \tag{12}$$

In the above, the Endpoint Lagrangian E has been introduced

$$E = G\left(\mathbf{s}(t_f), \mathbf{s}(0), t_f, t_0\right) + \boldsymbol{\nu}^\intercal \mathbf{e}\left(\mathbf{s}(t_f), t_f\right)\,.$$

E compiles all terminal cost and constraints conditions thanks to the set of terminal covectors $\boldsymbol{\nu}$, and play a symmetric role with respect to the Hamiltonian, whose focus is along the state trajectory and not its final values.

The rationale behind this work is the procedural, systematic approach of PMP towards solving any kind of optimal control problem and the special structure of its necessary conditions. In particular, the adjoint system in Eqs. (7) together with the Hamiltonian Evolution Condition Eq. (12) are just Hamilton's canonical equations applied to the problem's Hamiltonian. Thus, state coordinates and covectors play the same role as generalized coordinates and conjugate momenta do in Classical Mechanics,[20] with the caveat of the introduction of the control application $\mathbf{u}$ into the problem.

Furthermore, and specially interesting for this investigation, the adjoint system show a symplectic structure. Defining the vector $\mathbf{z} = [\mathbf{s}, \boldsymbol{\lambda}]^\intercal$, the adjoint equations can be compactly written as follows

$$\dot{\mathbf{z}} = \mathbb{J}\,\nabla_{\mathbf{z}} H\,, \quad \mathbb{J} = \begin{pmatrix} 0 & I_{n\times n} \\ -I_{n\times n} & 0 \end{pmatrix}\,.$$

$\mathbb{J}$ is the canonical symplectic matrix, which is, in the field of differential forms, a coordinate representation of a nondegenerate skew-symmetric billinear form Ω, mapping vectors and covectors to the real line

$$\Omega\left(\mathbf{a}, \mathbf{b}\right) = \mathbf{a}\,\mathbb{J}\,\mathbf{b}\,.$$

The symplectic nature of the adjoint system is one of the main technical difficulties to be overcome when solving PMP, as it introduces the curse of sensitivity: if the initial guess is not chosen in an educated manner, shooting algorithms are unstable and unreliable when solving the associated Two Boundary Value Problem to PMP.

5. Dataset generation

Before introducing the exact methodology used, a fundamental aspect in the design of supervised learning applications, such as classical neural network architecture implementations, is the generation of appropriate, meaningful datasets, from which the patterns of interest may be learned by the network. In most cases, this is the major bottleneck preventing the architectures from success.

When solving Optimal Control Problems, this task is even more challenging, as normally solving such problems in a mega-batch manner is computationally costly, and in most cases, optimality of the solutions cannot be totally proved, but through PMP conditions, precisely the step to be avoided in this work.

5.1 SBOP Solver

Instead of leveraging classical Optimal Control software, we resort into a novel, high-efficient algorithm, introduced in Solano-López et al.[16] and which has now reached a mature state. The technique is now briefly sketched for completeness purposes, as it plays a fundamental role in the success of the methods proposed hereafter. The special architecture of the solver provides a cost-efficient technique to solve Monte Carlo-like campaigns of optimal control problems, thus generating the needed datasets at nearly null expenses, when compared to standard optimization software.

The proposed algorithm is analogous to direct transcription and pseudospectral methods, in which the infinite dimensional Bolza problem is discretized into a Non-Linear Programming one, where the evaluation of the discrete cost

DOI: 10.13009/EUCASS2023-056

function is directly optimized. However, several aspects of the algorithm are particularised to provide a cost-efficient solving process.

The first step of the method consists in defining a discrete independent variable grid $\mathcal{T} = \{\tau_i\}_{i=0}^{N}$, at which the cost function, the path constraints and the boundary inequalities are evaluated. The specific distribution of points τ_i is not free, but coupled with another design aspect of the methodology, detailed now.

The fundamental characteristic of the method is the use of pre-determined functional bases to describe the state vector evolution, thus showing similarities with the classical family of shape-based solvers. In out method, the configuration (not the state) vector is projected onto some polynomial basis $\mathcal{P}$, which is free to be selected (although some families are recommended, such as that of Legendre or Chebyshev polynomials). Accordingly, the set $\mathcal{T}$ is selected to match the set of nodes of the family.

$$\begin{aligned} \mathcal{T} &= \{\tau_j \in \mathbb{R} \,|\, P_j(\tau) = 0\}\,; \\ \mathbf{s}(\tau_i) &= \sum_{j=0}^{N} \mathbf{c}_j \, P_j(\tau_i)\,. \end{aligned} \tag{13}$$

The polynomial order N may be different for each of the configuration vector components in a general formulation, nevertheless, for the rest of this work the contrary will be assumed to ease the notation. From the expansion of the configuration vector, derivatives are inmediately available.

$$\frac{\mathrm{d}^n}{\mathrm{d}\tau^n}\mathbf{s}(\tau_i) = \sum_{j=0}^{N} \mathbf{c}_j \, \frac{\mathrm{d}^n}{\mathrm{d}\tau^n} P_j(\tau_i)\,.$$

Once this is achieved, the dynamic constraints are intrinsically imposed by computing the control law $\mathbf{u}(t)$ as a residual to the state evolution derivatives, also known as a differential inclusion,[21] avoiding in this way the need of any integration, compared to classical direct methods. The control law $\mathbf{u}$ evaluated at the discrete grid $\mathcal{T}$ can be computed as the solution to

$$\mathbf{u}(\tau_i) \;:\; \dot{\mathbf{s}} - \mathbf{f}(\boldsymbol{\mu}, \mathbf{s}_i, \mathbf{u}_i, \tau_i) = \mathbf{0}\,. \tag{14}$$

The specific difference of our approach with respect to other methodologies is the inclusion of the boundary conditions in the prescribed functional/polynomial shape of the trajectory. Indeed, using such a semi analytic approach, the optimization solver only needs to handle path constraints. This is achieved by defining the symmetric set $\mathcal{B} = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_{N-1}, \mathbf{c}_N\}$ of polynomial coefficients and then fixing it to satisfy the boundary conditions through the following linear system:

$$\begin{aligned} \sum_{\mathbf{c}_j \in \mathcal{B}} \mathbf{c}_j \, P_j(0) &= \mathbf{s}_0 - \sum_{\mathbf{c}_j \notin \mathcal{B}} \mathbf{c}_j \, P_j(0)\,, \\ \sum_{\mathbf{c}_j \in \mathcal{B}} \mathbf{c}_j \, P_j(\tau_N) &= \mathbf{s}_f - \sum_{\mathbf{c}_j \notin \mathcal{B}} \mathbf{c}_j \, P_j(\tau_N)\,, \end{aligned} \tag{15}$$

The decision variable set to be optimized is therefore reduced to $\mathbf{x} = \{\mathbf{c}_j\}_{\mathbf{c}_j \notin \mathcal{B}}$ together with additional degrees of freedom, such as the time of flight. For Cauchy Two-Boundary Value Problems (TBVP), a minimum third-order polynomial expansion is needed to allow free degrees of freedom to be determined.

For a given TBVP problem of Bolza of the form of Eq. (1), the complete optimization procedure is summarized in the following scheme:

1. Define the discrete sampling grid $\mathcal{T}$ to match/sample the interval $[t_0, t_f]$. For time-free problems, a domain rescaling can be used to include the final time t_f as an optimization variable.
2. Generate an initial guess $\boldsymbol{\zeta}(\tau)$ for the optimal phase space flow using a densified sampling grid and Eq. (13) for a third-order expansion of the state vector, to analytically include boundary conditions in it.
3. Compute the initial guess for the decision variable $\mathbf{x}$ using a least-squares method on $\boldsymbol{\zeta}(\tau)$.
4. Optimize the decision variable vector $\mathbf{x}$ using an NLP solver. At each iteration,
   (a) Solve the set $\mathcal{B}$ through Eq. (15) to impose the boundary conditions.
   (b) Compute the control law using Eq. (14) at the sampling grid.
   (c) Minimize the cost function J in Eq. (1) over the sampling grid while respecting path constraints via NLP.
5. Once solved, obtain the optimal state evolution and the final control law through Eq. (13) and (14), respectively.

Standard NLP solver algorithms suffice to compute the solution of the problem of Bolza of interest via this approach, without the need of any initial guess. In this work, all examples have been computed using the Sequential-Quadratic Programming (SQP) algorithm, as built in Matlab's fmincon function.[22,23]

DOI: 10.13009/EUCASS2023-056

5.2 Dataset

Two different datasets are generated for further training of the neural network architecture, corresponding to the two problems proposed. In practice, given that our objective is the symbolic embedding of PMP into a deep learning architecture, distinctions between optimal control problems shall be disregarded, so as to prevent any bias in the final solution, inherent to the dataset. However, as this is just a preliminary step towards such goal, differentiation between the two problems will be partially maintained.

The standard 80-20% partition of the dataset allows for both training and test cases. Validation is handled internally by the training algorithms used.

LQR dataset In particular, 1000 3D LQR problems are randomly generated and solved via the SBOP technique. While a close-form solution exist, the numerical solution is preferred to reproduce the nominal both academic and industrial scenario, in which the former is not available.

The final conditions are specified for the 1000 problems, and correspond to strict regulation of the state vector

$$\mathbf{s}(t_f) = \mathbf{0}\,.$$

The initial conditions are however randomly generated to promote variance within the solution, sampled from a standard Gaussian distribution

$$\mathbf{s}_0 \sim \mathcal{N}(\mathbf{0}, I_{6\times 6})\ .$$

The realisation of the hyperparameters of the problem are particularised for the following weighting matrices, without any loss of generality

$$Q = I_{6\times 6}\,, \quad R = I_{3\times 3}\,.$$

The linear model (A, B) is also generated randomly, although maintained over all the solved cases

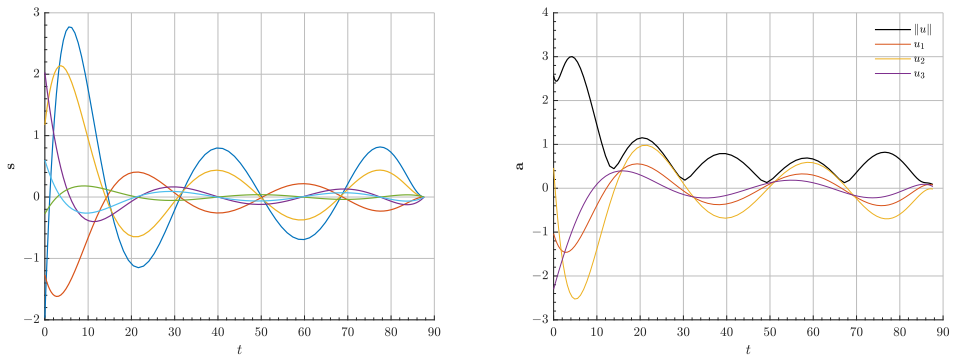
$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \quad A_1 = \begin{pmatrix} 0_{3\times 3} & I_{3\times 3}\,, \end{pmatrix} \quad A_2 = \begin{pmatrix} 0.4170 & 0.3023 & 0.1863 & 0.5388 & 0.2045 & 0.6705 \\ 0.7203 & 0.1468 & 0.3456 & 0.4192 & 0.8781 & 0.4173 \\ 0.0001 & 0.0923 & 0.3968 & 0.6852 & 0.0274 & 0.5587 \end{pmatrix}, \quad B = \begin{pmatrix} 0_{3\times 3} \\ I_{3\times 3} \end{pmatrix}.$$

The final time of flight is also sampled from a uniform distribution

$$t_f = 20 + 80 \cdot \mathcal{U}(0, 1)\,, \quad t_0 = 0\,.$$

The LQR trajectories are discretized into 100 evaluation points, following the distribution of the Legendre polynomials nodes.

An example of an optimal flow for the LQR, as solved by the proposed SBOP methodology, is depicted in Fig. 1.



(a) State trajectory solution for the LQR problem. (b) Control policy solution for the LQR problem.

Figure 1: Optimal state-control flow for one of the solved Monte-Carlo LQR problems.

DOI: 10.13009/EUCASS2023-056

Optimal asteroid landing As for the optimal landing problem dataset, given its intrinsic computational cost, only 100 randomly generated cases were solved via the SBOP method. All 3D trajectories are discretized into 20 evaluation points, following again the distribution of the Legendre polynomials nodes.

In this setup, the asteroid gravity field intensity κ and the maximum acceleration $\Gamma_{\max}$ are given by

$$\kappa = 1.6\,\mathrm{m/s^2}\,, \quad \Gamma_{\max} = 20\,\mathrm{m/s^2}\,.$$

To enhanced the numerical convergence of the algorithm, nonetheless, non-dimensional units will be used. In particular, the characteristic length l_c is taken to be the maximum ellipsoid semimajor axis.

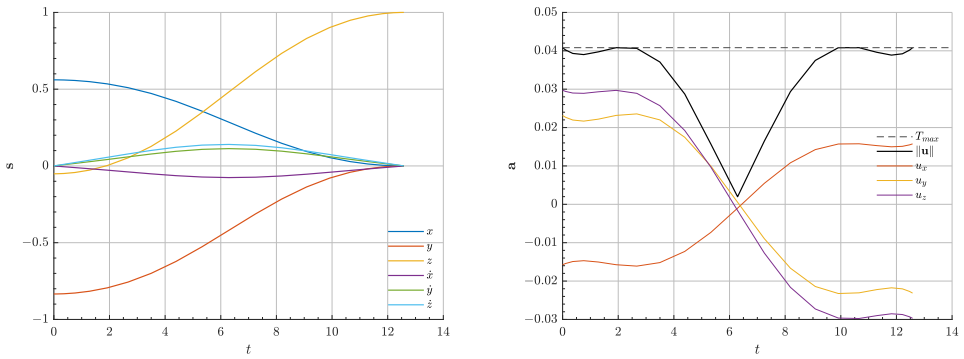
The final conditions are again fixed for all descent cases, reading

$$\mathbf{s}(t_f) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^\intercal\,,$$

which corresponds to landing on the asteroid North pole. The initial conditions are however sampled from the following Gaussian distribution

$$\begin{aligned} \mathbf{s}_0 &= \begin{bmatrix} \mathbf{r}_0 & 0 & 0 & 0 \end{bmatrix}^\intercal\,, \\ \mathbf{r}_0 &\sim \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\intercal + \mathcal{N}(\boldsymbol{\mu}, \Sigma)\,, \\ \boldsymbol{\mu} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\intercal\,, &\quad \Sigma = \mathrm{diag}\,(0.1,\, 0.1,\, 1)\,. \end{aligned}$$

One particular solution for a randomly-generated optimal landed problem, again, solved by the proposed SBOP methodology, is shown in Fig. 2.



(a) State trajectory solution (canonical units).

(b) Control policy solution (canonical units).

Figure 2: Optimal state-control flow for one of the solved Monte-Carlo optimal landing problems.

5.3 Data augmentation

The use of the SBOP solver comes with additional advantages to the generation of machine learning datasets. Mainly, because the state evolution is returned as a linear functional decomposition C on a given polynomial basis P_J, it can be re-sampled on an overdensified independent grid $\mathcal{D}$, thus providing additional samples to the original ones for which the problem was solved, but at no computational expenses. These data augmentation procedure shall be performed over the training dataset only, while testing only applies for the original test solutions set.

In short, after computing the solution matrix C over the original node grid $\mathcal{T}$, a new grid $\mathcal{D}$ can be defined to evaluate the very same optimal state evolution

$$\mathbf{s}\,(\tau_j) \to \mathbf{s} = \sum_{i=0}^{N} \mathbf{c}_i\, P_i\,(\tilde{\tau}_j)\,, \quad \tau_j \in \mathcal{T},\, \tilde{\tau}_j \in \mathcal{D}\,, \quad \mathrm{card}(\mathcal{D}) > \mathrm{card}(\mathcal{T})\,.$$

Once the new sampled trajectory is obtained, the differential inclusion Eqs. 14 can be evaluated again to obtain the corresponding control law, thus completing the tuple associated to the optimal flow of the original problem $\left[\mathbf{s}(\tau_j),\, \mathbf{u}(\tau_j)\right]$.

In fact, infinite data may be obtained from each single solution, which, in spite of corresponding to the same problem (in terms of boundary conditions, path constraints...), provide a realisation of the inherent functional policy between the state evolution and the control law $\mathbf{u}$.

DOI: 10.13009/EUCASS2023-056

This data augmentation technique shall be conceptually compared to previous examples of the very same purpose. In our case, this process is a much simpler exercise than those found in the literature, as it does not require further solving of additional optimization or propagation problems,[12] with the associated computational burden, but just re-evaluation of an analytical solution.

However, the path constraints are not strictly satisfied on these new samples and thus they cannot be technically consider a strict solution to the problem. Nonetheless, if the original problem is solved with accuracy, this will not be a real, practical concern, especially when considering it for data augmentation techniques. Any bias of the solution and its effect on the performance will be analysed within the testing campaign.

6. Methodology and network architecture

The two novelties of this work are technically introduced in this Section, in which the classical direct/indirect approaches for solving Optimal Control problems are transformed to an analogous partition within general regression of control policies from state trajectories.

6.1 Direct regression problem

The direct regression problem provides the deep learning counterpart of the direct optimization approach in Optimal Control, in which the solution flow $(\mathbf{s}^*, \mathbf{u}^*)$ is obtained via direct discretization (the dataset) without invoking any formulation in the co-state space, typical of PMP and the indirect approach. Thus, the optimal control problem is then transformed into a regression one: given the state trajectory and the instantaneous running cost, the optimal control policy shall be regressed by the ML architecture to minimize the desired performance index and fulfil the final desired state. While much simpler, the focus of NN within control optimization has mainly been put into estimation of the covectors of the problem, a much more challenging task to which this Section is also devoted.

To achieve such regression, Generalized Regression Neural Networks (GNRR) are used as the main DL architecture, given their intrinsic, dedicated design to problems of this type.

GNRR were first suggested by Specht in 1991 and generalize Radial Basis Function Networks.[24] GRNNs fall into the category of probabilistic neural networks and implement their learning algorithm through nonparametric estimators of probability density functions. They are also universal estimators commonly used for regression-like problems, due to their ease of design and implementation and high performance in converging to the nonlinear regression surface. However, they may require unbounded sizes to do so.

GRNNs are composed of only two hidden layers, as seen in Fig. 3. The input layer connects to normalized radial basis neurons, the pattern layer, followed by a special linear or summation layer

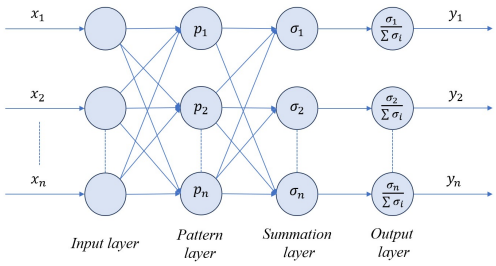


Figure 3: Canonical Generalized Regression Neural Network architecture.

Radial basis neurons, when activated, reproduce the following output $\mathbf{y}$

$$\mathbf{y} = \exp\left(\beta \|\mathbf{x} - \mathbf{c}\|_2\right) ,$$

where $\mathbf{x}$ is the neuron input vector, β is known as the bias and $\mathbf{c}$ as center or weight vector of the Gaussian probability distribution. Differently from other architectures, GRNNs are trained (their parameter set optimized) in a single pass, not using backpropagation, but nonparametric estimation: the input-output metric is substituted by a Gaussian kernel taking the Euclidean distance $\|\mathbf{x} - \mathbf{c}\|$ as its argument. Other probability distribution functions may also be used. The estimation of β is achieved through standard least-squares methods in the final output layer with respect to the expected output in the dataset, which may be sparser than needed when using other architectures.

Regression and training of the network is implemented after the data augmentation process for the training set, in which each optimal flow (given by the coefficient matrix C) is further sampled by a factor of 50 with respect to the original number of independent grid nodes.

DOI: 10.13009/EUCASS2023-056

The exact GNRR architecture used is commercially provided by the software Matlab, in its 2021b implementation.[25] As for its input and ouptus, these are defined to be

$$\mathbf{x} = \left[\tau_i \quad \mathbf{s}_0 \quad \mathbf{s}_f \quad \mathbf{s}_i^* \quad \int_{t_0}^{t_i} l_i^* \, \mathrm{d}t\right]^\intercal , \quad \mathbf{y} = \mathbf{u}^* .$$

6.2 Indirect regression problem

The indirect approach approaches the very same regression problem (estimation of the control policy from the state trajectory and the running cost) by means of a chain of simple feedforward neural networks, which are symbolically represented by transfer functions $\mathcal{P}_i$.

The first neural network is input the current time, state trajectory vector, initial and desired final conditions and running cost $\mathbf{x} = [\tau_i, \mathbf{s}_0, \mathbf{s}_f, \mathbf{s}_i, l_i]^\intercal$. After the composition of the two neural networks, the optimal control policy $\mathbf{u}$ is recovered.

$$\mathbf{u} = \mathcal{P}_2 \circ \mathcal{P}_1 \circ \mathbf{x} .$$

Thus, the complete function $\mathbf{f} = \mathcal{P}_2 \circ \mathcal{P}_1$ may be thought of as a classical nonlinear feedback controller.

The use of a two chained NN can be thought of as an autoencoder-like structure, although in our case compression is substituted by actually decompression after $\mathcal{P}_1$ (augmentation of the space dimensionality). It provides an intermediate compression step towards $\mathbf{u}$ and additional neuron connections to accurately recover the needed control policy $\mathbf{f}$.

The specific output of $\mathcal{P}_1$ is leveraged to impose PMP necessary conditions within the regression process. A quick analysis of these PMP conditions reveals that the co-states $\boldsymbol{\lambda}$ play a fundamental role in all of them, and their computation appears to be inevitable when properly addressing Optimal Control. Therefore, to introduce them into the solver, the output of $\mathcal{P}_1$ is matched to be the symplectic vector $\mathbf{z}$

$$\mathbf{y}_1 = \mathcal{P}_1 \circ \mathbf{x} = \begin{bmatrix} \mathbf{s} \\ \boldsymbol{\lambda} \end{bmatrix} .$$

In fact, in this way, $\mathbf{f}$ aims to capture the resulting policy from PMP

$$\mathbf{u} = \mathbf{f}\,(\mathbf{s}, \boldsymbol{\lambda}, l) \ ,$$

in which $\boldsymbol{\lambda}$ are the hidden variables of the problem.

The exact embedding of PMP within $\mathcal{P}_1$ is achieved via its particular symplectic structure. In particular, the following regularization term is used as a measure of symplecticity of the output of $\mathcal{P}_1$

$$\delta\Omega = \left(\mathbf{z}_T^\intercal \, \mathbb{J} \, \mathbf{z}_y\right)^2 .$$

which admits the interpretation of a weighted least-squares penalty term. Because the 2-form Ω is non-degenerate, it follows that

$$\mathbf{v} \, \mathbb{J} \, \mathbf{v} = 0 .$$

For symplectic systems, the above results appears to be a more natural similarity metric than the traditional MSE. In the above, $\mathbf{z}_T$ refers to the symplectic target output and $\mathbf{z}_y$ to the true network outcome.

In this way, the standard Euclidean loss function is modified as follows

$$J = \|\,\mathbf{x} - \mathbf{y}\|_2^2 \rightarrow J = \|\mathbf{x} - \mathbf{y}\|_2^2 + \gamma \, (\mathbf{y} \, \mathbb{J} \, \mathbf{x})^2 \ .$$

In the above, $\gamma > 0$ is a penalty parameter and $\mathbf{x}$ and $\mathbf{y}$ represent the network input and output, respectively.

The inclusion of the regularization term allows to promote the symplecticity of the solution, which is fundamentally required by PMP. This strategy for enhanced performance is analogous to that found for the so-called physics-informed neural networks, in which abstract physics conservation laws (such as energy or momentum conservation) are embedded into the architecture training.[17] This promotion of the symplecticity of solutions also avoids more complex architectures, such as those proposed in,[15] which require of auto-differentiation routines among others.

7. Applications and results

The following Section presents some verification scenarios in which the novelties presented in the above are compared against classical deep learning solutions for optimal control. Results are however preliminary and shall be refined in the upcoming months. An update version of this paper will be published soon elsewhere.

All simulations were performed using an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz with 15.7 GB of RAM. The physical architecture of the machine used is the major performance bottleneck as for June 2023.

DOI: 10.13009/EUCASS2023-056

7.1 Solution and results for the direct problem

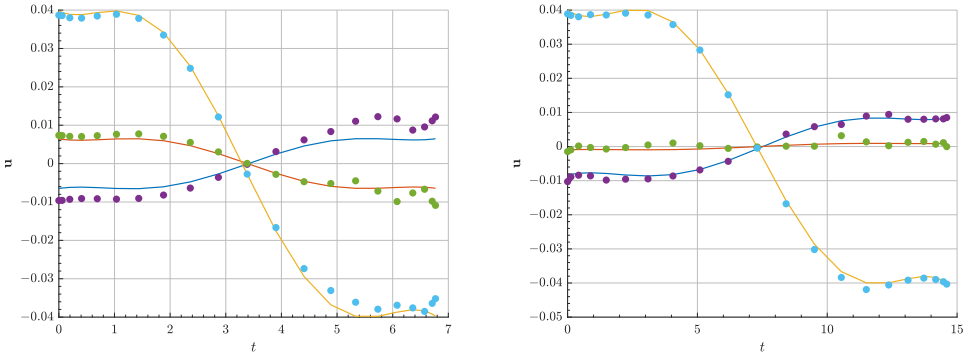
First, the performance of the simple GNRR architecture with respect to a standard feedforward (FF) network over the optimal landing dataset is addressed (given its scarcity), without any additional technique, such as data augmentation, thus providing a worst-case like scenario. In particular, the feedforward network is constructed upon 4 hidden layers of 20 neurons each, as detailed in Solano-López,[14] providing a medium-size architecture.

Table 1 shows the associated performance indices to the regression, training and test process. These proposed performance indices aim to characterize the computational cost of the process, and the final Root Mean Squared Error (RMSE) over the training set, for each optimal trajectory and across all of it. As it can be observed, the computational cost of using the GNRR is comparatively null to that of the FF architecture at the exchange of a really small penalty of regression performance.

Table 1: Main performance indices for the GNRR and FF architectures.

	Optimal landing - GNRR	Optimal landing - FF
Computational time [s]	0.2163	49.7306
Mean RMSE	7.6441×10^{-5}	1.1889×10^{-5}
3σ RMSE	2.6150×10^{-4}	7.5176×10^{-5}

Fig. 4 graphically shows the typical reconstruction performance of the test control law by the two networks against its true value.



(a) Regression example of the optimal control policy using GNRR.

(b) Regression example of the optimal control policy using FF.

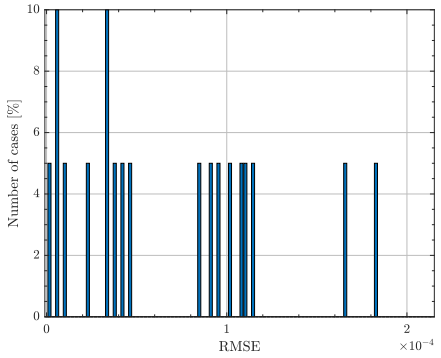
Figure 4: Typical regressed solutions for the direct problem using the two architectures.

Once the increased performance in the efficiency index computational cost over final regression error of the GNRR architecture has been demonstrated, the very same problem is again solved to further explore the numerical advantages of the methods proposed. Indeed, in this case, data augmentation is used to provide extended synthetic information for the training set. The solution results can be analysed in Table 3 and the error distributions in Fig. 6, in which the effect of the augmented dataset can be appreciated.

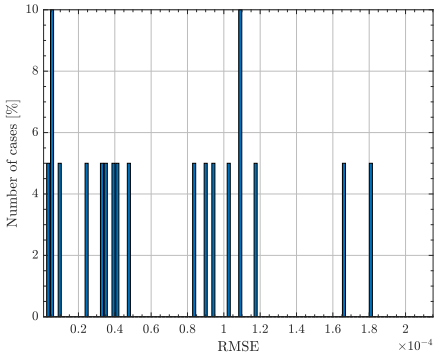
Table 2: Main performance indices for the GNRR solution with and without data augmentation.

	Optimal landing - augmented	Optimal landing - no augmentation
Computational time [s]	0.2177	0.2163
Mean RMSE	7.6250×10^{-5}	7.6441×10^{-5}
3σ RMSE	2.6167×10^{-4}	2.6150×10^{-4}

DOI: 10.13009/EUCASS2023-056



(a) Error distribution for the augmented dataset using GNRR.



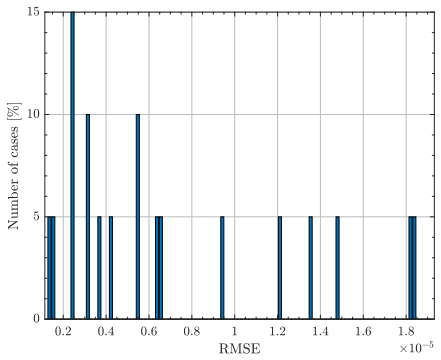
(b) Error distribution for the original dataset using GNRR.

Figure 5: Effect on the synthetic augmented training in the final error distributions.

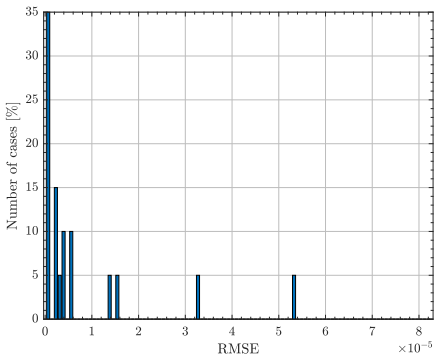
Using the FF network instead, the following results are obtained

Table 3: Main performance indices for the FF solution with and without data augmentation.

	Optimal landing - augmented	Optimal landing - no augmentation
Computational time [s]	893	49.7306
Mean RMSE	7.7704×10^{-6}	1.1889×10^{-5}
3σ RMSE	2.6030×10^{-5}	7.5176×10^{-5}



(a) Error distribution for the augmented dataset using FF.



(b) Error distribution for the original dataset using FF.

Figure 6: Effect on the synthetic augmented training in the final error distributions.

The results are actually insightful, and deserve appropriate analysis. The GNRR architecture appears to be insensitive to the use of data augmentation, due to its architecture, which do not benefit that much from the size of the dataset but rather on the variance and information contained in the dataset. For the same number of solved problems used in the training, similar results will be always obtained. Still, the data augmentation technique is able to skew the error distribution towards more accurate solutions. However, this improvements are nearly negligible when compared to the FF network. In this case, the errors diminish by an order of magnitude, and reduces to nearly half for the 3σ case, therefore validating our proposal for data augmentation. In fact, the performance over the testset is significantly restricted to a much smaller error interval, featuring a reduction of a factor of 10 in its dimension. This is explained through the use of backpropagation in the FF network. On the other hand, obviously, the computational time also increases by an order of magnitude.

DOI: 10.13009/EUCASS2023-056

7.2 Solution and results for the indirect problem

As for the indirect regression problem, a similar comparison is now addressed to that found previously, in which the performance of the novel architecture is compared against classical regression solutions. In particular, the direct formulation will be compared against its indirect counterpart: introducing the network partition $\mathcal{P}_1$ into the regression problem against directly optimizing for $\mathbf{u}$ given $\mathbf{s}$, thus avoiding any dual or PMP-embedding step.

First, for the LQR dataset, the co-states are obtained via the analytical solution to the problem, but through the use of the numerical solutions obtained, similarly to the case of other pseudospectral solvers.[9]

$$\boldsymbol{\lambda}_j = S(\tau_j) \sum_{i=0}^{N} \mathbf{c}_i P_i(\tau_j) .$$

Similarly, the derivatives of the co-vectors are also obtained via

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{\lambda}_j = \frac{\mathrm{d}}{\mathrm{d}\tau}S(\tau_j) \sum_{i=0}^{N} \mathbf{c}_i P_i(\tau_j) + S(\tau_j) \sum_{i=0}^{N} \mathbf{c}_i \frac{\mathrm{d}}{\mathrm{d}\tau} P_i(\tau_j) .$$

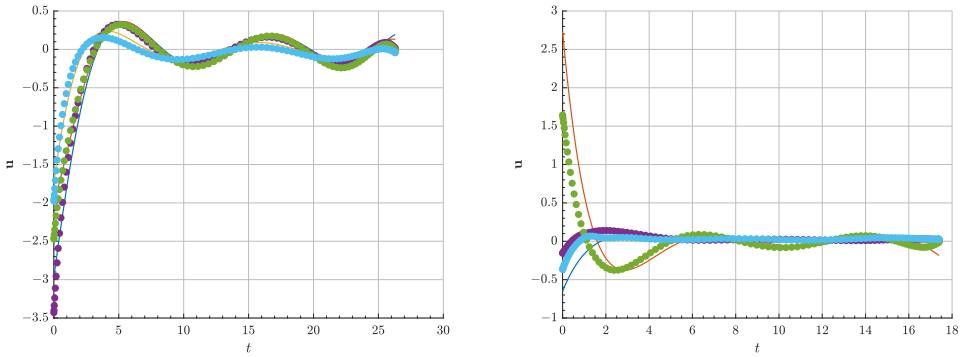
The performance of the direct method over this particular regression problem is summarized in Tab. 4, in which the effect of including the symplectic regularizing term is also shown. In both cases, the same neural network architecture is used, given by a fully-connected feedforward net of 10 neurons per layer and 2 hidden layers, featuring a really small network. For the indirect formulation, however, the training subset is restricted to be one third of the original set, featuring only 300 solved problems, so as to proof the generability of the resulting network architecture.

Moreover, in the training of $\mathcal{P}_2$, the weights of $\mathcal{P}_1$ can be used to initialize the architecture of the former, in an example of transfer learning.

Table 4: Main performance indices for the FF solution with and without symplectic regularization.

	LQR - direct	LQR - regularized
Computational time [h]	1.5	0.25
Mean RMSE	0.1194	0.1394
3σ RMSE	2.6012	1.6356

Fig. 7 graphically shows the typical reconstruction performance of the test control law by the two network training algorithms against its true value. Moreover, the final error distribution is also given in Fig. 8.



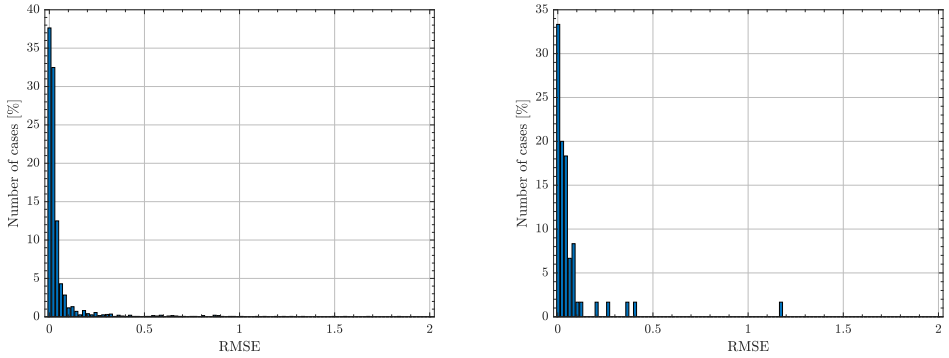
(a) Regression example of the optimal control policy using the direct method.

(b) Regression example of the optimal control policy using the indirect method.

Figure 7: Typical performance of the direct and indirect formulations in the regression problem.

The results, albeit the differences in the training set, are really similar in terms of performance, both in mean RMS and the 3σ figure. Obviously, the training time is reduced by a factor of 5 in the symplectic case. This is explained partly by the reduced training dataset. However, when compared to the direct case, the $\mathcal{P}_2$ network only needed 20 iterations before validation occurred, when compared to the 1000 of the direct case, which also reduces the computational load of the process (despite the need to double the number of training process). The $\mathcal{P}_1$ appears to

DOI: 10.13009/EUCASS2023-056



(a) Error distribution for the direct problem. (b) Error distribution for the indirect problem.

Figure 8: Effect of the symplectic regularization in the regression solution.

perform well as a pre-processor for $\mathcal{P}_2$. And most importantly, the results generalize to similar performance in both cases, despite the difference in the size of the training batch. While this may relate to overfitting, none of the results appear to be explained by that, comparing both error distributions, Fig. 8. All in all, the indirect architecture shows promising results, providing a Proof of Concept of the proposed design, but which also deserve more in-depth analyses and further refinement.

8. Conclusions

This work proposes novel Neural Network training algorithms and architectures to solve with low-cost general Optimal Control problems. In particular, data augmentation and the Hamiltonian structure of Optimal Control (given by the application of Pontryagin's Maximum Principle) is leveraged to modify physics-informed NN and other standard deep learning architectures to the control practitioner's advantage. This HMC embedding is used to construct and train the network in a robust and generalizable manner.

In short, the training algorithm and the network architecture are designed to capture the Hamiltonian minimization conditions arising from PMP, which corresponds to the necessary optimality conditions for the general problem of Bolza. While classical deep learning applications are oriented towards presenting universal approximation functions to the datasets of interest, this work instead tries to capture high-level functional relationships, applicable to all optimal control problems. This is achieved via two novelties: first, a novel cost-effective, pseudospectral-like optimal control solver is used for low-cost data augmentation of optimal state-control tuple trajectories, and then combined with General Regression Neural Networks to solve the so-called direct regression problem. On the other hand, a chained feedforward-like architecture is used to provide the very same state-control trajectory via estimation of the co-states, through a physics-informed like, regularized loss function, in which symplecticity of the solution is promoted. These two new algorithms are compared to traditional deep learning architectures to assess their performance via preliminary simulation test results.

The scope of this study is however restricted in this first communication to particular applications to the classical LQR problem and that of optimal landing on an asteroid, in which the performance of the proposed architectures are compared to traditional deep learning approaches for optimal control regression and embedding. Obviously, the removal of this bias in the results remains as an open line of research, with the ultimate goal of developing a sufficiently general architecture so as to avoid any distinction between optimal control problems. Among all open lines of research, the development of an architecture able to capture the co-state evolution from the state trajectory is one of the primary goals to be achieved.

9. Acknowledgments

S.C.d.V., H.U. and P.S.-L. wish to acknowledge the Spanish State Research Agency for their support through the research grants PID2020-112576GB-C22 funded by MCIN/AEI/ 10.13039/501100011033, and TED2021-132099B-C32 funded by MCIN/AEI/ 10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR".

DOI: 10.13009/EUCASS2023-056

References

[1] S. Cuevas del Valle, B. Jiménez, C.D. Moreno, and M. Renieblas. Deep learning architectures for global operation and control of minituarized satellite constellations. In *Proceedings of the 73rd International Astronautical Congress (IAC)*. 2022.

[2] D. Izzo, M. Märtens, and B. Pan. A survey on artificial intelligence trends in spacecraft guidance dynamics and control. 12 2018.

[3] H. Li, H. Baoyin, and F. Topputo. Neural networks in time-optimal low-thrust interplanetary transfers. *IEEE Access*, 7:156413–156419, 2019.

[4] N.B. LaFarge, K.C. Howell, and D.C Folta. An autonomous stationkeeping strategy for multi-body orbits leveraging reinforcement learning. In *AIAA SCITECH 2022 Forum*, page 1764, 2022.

[5] Carlos Sánchez-Sánchez and Dario Izzo. Real-time optimal control via deep neural networks: Study on landing problems. *Journal of Guidance, Control, and Dynamics*, 41, 10 2016.

[6] L. Cheng, H. Li, Z. Wang, and F. Jiang. Fast solution continuation of time-optimal asteroid landing trajectories using deep neural networks. *Acta Astronautica*, 167, 11 2019.

[7] G.V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[8] I.M. Ross and M. Karpenko. A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2):182–197, 2012.

[9] I.M. Ross. *A Primer on Pontryagin's Principle in Optimal Control*. 2015.

[10] B. Conway. A survey of methods available for the numerical optimization of continuous dynamic systems. *Journal of Optimization Theory and Applications*, 152:271–306, 02 2012.

[11] T. Nakamura-Zimmerer, Q. Gong, and W. Kang. Neural network optimal feedback control with enhanced closed loop stability. In *2022 American Control Conference (ACC)*, pages 2373–2378, 2022.

[12] D. Izzo and S. Origer. Neural representation of a time optimal, constant acceleration rendezvous. *Acta Astronautica*, 08 2022.

[13] D. Izzo, E. Blazquez, R. Ferede, S. Origer, C. De Wagter, and G. Croon. Optimality principles in spacecraft neural guidance and control. 05 2023.

[14] P. Solano-López, A. Barea, R. Gutierrez-Ramon, and H. Urrutxua. Assessment of machine learning techniques for time-optimal landing trajectories. In *Proceedings of the 9th European Conference For Aeronautics And Space Sciences (EUCASS)*. 2022.

[15] T. Meng, Z. Zhang, J. Darbon, and G. Karniadakis. Sympocnet: Solving optimal control problems with applications to high-dimensional multiagent path planning problems. *SIAM Journal on Scientific Computing*, 44(6):B1341–B1368, 2022.

[16] P. Solano-López, S. Cuevas del Valle, A. Solá, and H. Urrutxua. An investigation on shape-based methods with different polynomial basis for low-thrust mission design. In *Proceedings of the 9th European Conference For Aeronautics And Space Sciences (EUCASS)*. 2022.

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

[18] R. Bellman. *Dynamic Programming*. Dover Books on Computer Science, 1957.

[19] I.M. Ross. Space trajectory optimization and l1-optimal control problems. In P. Gurfil, editor, *Modern Astrodynamics*, volume 1 of *Elsevier Astrodynamics Series*, pages 155–VIII. Butterworth-Heinemann, 2006.

[20] T. Lee, M. Leok, and N.H. McClamroch. *Global formulations of Langragian and Hamiltonian Dynamics on Manifolds*. Springer, 2018.

DOI: 10.13009/EUCASS2023-056

[21] F. Fahroo and I. Ross. Second look at approximating differential inclusions. *Journal of Guidance Control and Dynamics*, 24:131–133, 01 2001.

[22] MathWorks. fmincon. `https://es.mathworks.com/help/optim/ug/fmincon.html`. Online; accessed 04 July 2023.

[23] J.T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Springer, 2010.

[24] D.F. Specht. A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576, 1991.

[25] MathWorks. Generalized regression neural networks. `https://es.mathworks.com/help/deeplearning/ug/generalized-regression-neural-networks.html`. Online; accessed 04 July 2023.