


RESEARCH ARTICLE

Open Access



# Optimal trajectory planning combining model-based and data-driven hybrid approaches

Chady Ghnatios<sup>1\*</sup>, Daniele Di Lorenzo<sup>2</sup>, Victor Champaney<sup>2</sup>, Amine Ammar<sup>3,4</sup>, Elias Cueto<sup>5</sup>  and Francisco Chinesta<sup>2,4</sup>

\*Correspondence:  
Chady.Ghnatios@ensam.eu

<sup>1</sup>PIMM Lab & SKF Chair, Arts et Metiers Institute of Technology, Paris, France

<sup>2</sup>PIMM Lab & ESI Chair, Arts et Metiers Institute of Technology, Paris, France

<sup>3</sup>LAMPA Lab & ESI Chair, Arts et Metiers Institute of Technology, Angers, France

<sup>4</sup>CNRS@CREATE CNRS, Singapore, Singapore

<sup>5</sup>ESI Group Chair, Aragon Institute of Engineering Research, Universidad de Zaragoza, Zaragoza, Spain

## Abstract

Trajectory planning aims at computing an optimal trajectory through the minimization of a cost function. This paper considers four different scenarios: (i) the first concerns a given trajectory on which a cost function is minimized by acting on the velocity along it; (ii) the second considers trajectories expressed parametrically, from which an optimal path and the velocity along it are computed; (iii), the case in which only the departure and arrival points of the trajectory are known, and the optimal path must be determined; and finally, (iv) the case involving uncertainty in the environment in which the trajectory operates. When the considered cost functions are expressed analytically, the application of Euler–Lagrange equations constitutes an appealing option. However, in many applications, complex cost functions are learned by using black-box machine learning techniques, for instance deep neural networks. In such cases, a neural approach of the trajectory planning becomes an appealing alternative. Different numerical experiments will serve to illustrate the potential of the proposed methodologies on some selected use cases.

**Keywords:** Machine learning, Artificial intelligence, Optimal planning, Trajectory optimization, Euler–Lagrange equations, Physics informed machine learning

## Introduction

Optimal trajectory planning is a topic present in many domains of science, engineering and technology. To cite a few examples, one has to determine the optimal velocity of an autonomous car moving along a prescribed path for example, the optimal path of a drone, or the optimal deposition trajectory in additive manufacturing and 3D printing processes.

In some cases, the trajectory is known, but the optimal conditions when moving along the path must be defined to minimize a given cost function. When addressing autonomous vehicles, for instance, fuel consumption must be minimized, with direct consequences on the emission reduction [1,2]. Many works focus on the definition of an adequate cost function that should reflect a compromise between the energy consumption and the travel time. Once such a cost function is defined, different techniques exist to determine the optimal conditions all along the path [3–7]. Even if reinforced learning [8] seems

particularly well adapted for that purpose [9], the authors proposed recently an alternative route by making use of neural network-based regressions informed by the Euler–Lagrange equation [10].

The present paper proposes an extension of the just referred work by considering more general situations in which trajectories could be expressed parametrically, or they will result from the optimality conditions. It also proposes a valuable framework for addressing problems involving a noticeable degree of uncertainty or variability. All the resulting procedures will be deeply discussed.

The optimal planning problem becomes more tricky when the trajectories are expressed parametrically, or even more, when the trajectory could, and should, be obtained with the only knowledge of the departure (at the initial time) and the arrival point (at the expected arrival time). The situation becomes even more complex when the environmental conditions exhibit unpredictable fluctuations, only expressible in a statistical sense.

All these scenarios can be stated as optimization problems, which have been extensively addressed by using standard numerical optimization techniques. Even if using these procedures implies a subsequent computational cost, in general, trajectory planning has traditionally been performed offline, by assuming forecasted environmental conditions involved in the cost functions. However, engineering applications are moving fast from offline procedures to online applications. Digital twins are peopling all the domains of engineering and technology. In real-life operation applications, a digital twin of an agent or an asset, must react in real-time to any environmental change. For instance the change of the wind orientation or speed when operating a drone, or the sea current for a ship.

Nowadays, solutions of parametrized physics-based models including environmental conditions (e.g., temperature maps, wind map, current map, electromagnetic maps,...) can be obtained by using the most recent techniques of meta-modeling for the construction of the associated surrogate models [11–13]. Thus, the environmental conditions evolving in time can be evaluated globally in almost real-time, and then, optimizations involved in trajectory planning should perform online, pushing the available standard techniques to their limits.

It is here that artificial intelligence and machine learning methods come into play [14]. In order to speed-up optimal decisions, reinforcement learning is increasingly considered, as commented before, in particular for autonomous system applications [9]. However, fully data-based techniques face many difficulties: (i) accomplishing a proper and complete training; (ii) the risks when operating outside the training region (out-of-distribution regime, extrapolation); and (iii) the difficulty of explaining decisions (black-box procedures).

In mechanical engineering, many optimization problems can be efficiently solved by minimizing a global cost function, that results from the integral of a local cost along a path. The global cost optimality conditions result in the so-called Euler–Lagrange equations. Thus, parametric cost functions can lead to parametrized Euler–Lagrange equations, enabling the construction of parametric optimal trajectories. Such an approach is extremely valuable for driving an autonomous asset in operation, as discussed later in this paper.

However, sometimes cost functions are not explicit. Thus, it becomes difficult to operate with them by applying the Euler–Lagrange equation. For instance, when the cost function is obtained by testing the asset under several operating scenarios, and then, expressed

from a deep neural network. In such a case, the Euler–Lagrange equation must be solved following an unusual procedure, as detailed later in this work. Physics Informed Neural Networks (PINNs) offer an appealing gateway for performing in these settings [15].

This work considers the four aforementioned scenarios. The first concerns a given trajectory on which a cost function is minimized. This is addressed in “[Optimality along a given trajectory](#)”. The second case, which tackles the case of parametric trajectories and the extraction of the optimal path through parameter optimization, is discussed in “[Optimal planning on parametrized trajectories](#)”. “[Optimal trajectory and planning](#)” addresses the case in which only the departure and arrival points of the trajectory are known, and the optimal path (in the sense of minimizing a given cost function) is to be determined. Finally, “[Future prospects: operating in stochastic environments](#)” proposes a methodology to extend the approach to stochastic settings, putting in perspective its integration in complex paths planners.

### Optimality along a given trajectory

We assume a cost function  $\mathcal{C}$ , defined as a function of the position  $s$ , the curvilinear coordinate along a given trajectory  $\Gamma$ , and as a function of the (module of) velocity  $v(s)$  and its variation with respect to the curvilinear coordinate  $s$ ,  $v'(s) = dv/ds$ . The dependence on  $v(s)$  accounts for the drag and  $v'(s)$  will be used to penalize the change of the velocity along the trajectory [10]. The change of velocity entails a cost because of the inertia. However, the considered formulation that penalizes the change of velocity with respect to the curvilinear coordinate becomes simpler than the one making use of the acceleration (velocity time derivative). This point is of particular interest when the environmental conditions are fluctuating fast, and we should avoid that the agent must adapt to its environment continuously by modifying its velocity. In those situations the use of the penalty term suffices for obtaining an adequate response. Thus, the local cost  $\mathcal{C}$  takes the form  $\mathcal{C}(s, v(s), v'(s))$ . The environmental conditions entering the cost function will be described later.

Now, denoting by  $S$  the total path length, the total cost associated with the trajectory  $\Gamma$ ,  $\mathcal{I}_\Gamma$ , can be obtained using the integral

$$\mathcal{I}_\Gamma = \int_0^S \mathcal{C}(s, v(s), v'(s)) ds. \quad (1)$$

The optimization problem looks for the optimal velocity along the path  $\Gamma$ ,  $v^{\text{opt}}(s)$ , by minimizing the functional  $\mathcal{I}_\Gamma$  with prescribed initial and final velocities,  $v(0)$  and  $v(S)$ , respectively. This minimization becomes equivalent to the fulfillment of the Euler–Lagrange (EL) equation:

$$\frac{\partial \mathcal{C}}{\partial v} - \frac{d}{ds} \frac{\partial \mathcal{C}}{\partial v'} = 0. \quad (2)$$

When the cost function is expressed analytically as a function of  $s$ ,  $v(s)$  and  $v'(s)$ , the EL equation results in a differential equation involving the variable  $v(s)$ , whose solution results in the optimal velocity  $v^{\text{opt}}(s)$ .

### Cost functions expressed analytically

Let us consider, for example, the instantaneous cost function given by  $\mathcal{C} = \frac{1}{2}v^2(s) + \frac{\beta}{2}(v(s) - \bar{v})^2 + \frac{1}{2}v'^2(s)$ , where  $\beta$  is the weight enforcing the target velocity  $\bar{v}$ . The optimal

velocity results from the EL equation solution, which, in the present case, takes the form:

$$\frac{\partial \mathcal{C}}{\partial v} = v + \beta(v - \bar{v}), \quad (3)$$

and

$$\frac{d}{ds} \frac{\partial \mathcal{C}}{\partial v'} = v'', \quad (4)$$

which result in the following ODE:

$$(1 + \beta)v(s) - \beta\bar{v} - v'' = 0. \quad (5)$$

The integration of Eq. (5) requires two boundary conditions. In this example, we consider vanishing initial and final velocities,  $v(s = 0) = v(s = S) = 0$ . Solving the ODE problem with the appropriate boundary conditions results in the optimal velocity  $v^{\text{opt}}(s)$  minimizing the total cost. Such a solution can be obtained analytically or numerically, by using the finite difference method for example.

In the case of neglecting the term involving  $v'(s)$ , the cost function reduces to the algebraic equation  $\mathcal{C} = \frac{1}{2}v^2(s) + \frac{\beta}{2}(v(s) - \bar{v})^2$ , and the optimal velocity can be trivially obtained from:

$$v(s) \equiv v^{\text{opt}}(s) = \frac{\beta}{1 + \beta}\bar{v}. \quad (6)$$

In Eq. (6), when  $\beta \rightarrow \infty$ ,  $v^{\text{opt}}(s) = \bar{v}$ , and when  $\beta \rightarrow 0$ , without any constraint to perform the travel, the best choice is stay at rest to minimize the cost induced by the drag, i.e.,  $v^{\text{opt}} = 0$ .

### Cost functions expressed from black-box oracles

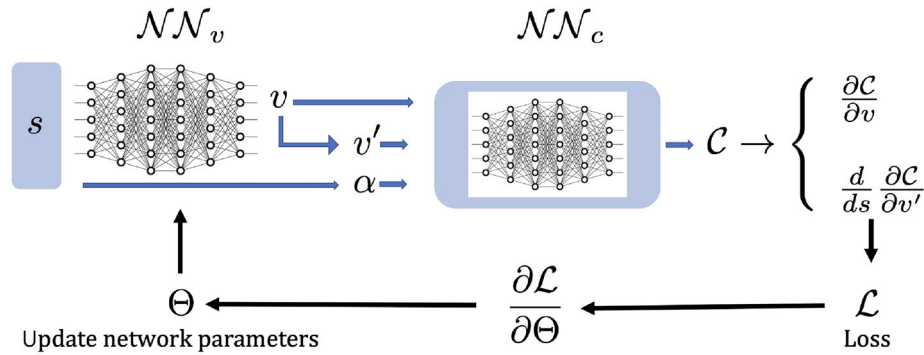
When the cost function is represented by a trained black-box, a sort of oracle—a deep neural network (DNN) for instance—the just described procedure can not be applied, as the explicit form of the ODE function can not be obtained.

In our recent work [10], we proposed the use of a neural network able to perform the regression of an expression of the type  $v = \mathcal{N}_v(s)$ . Then, automatic differentiation allowed us to compute  $v'(s)$ . The velocity and its derivative, as well as the local conditions affecting the movement (for example the local wind in the case of a flight of a drone) at position  $s$ , and expressed by the term  $\alpha(s)$ , are the inputs of the cost function neural network  $\mathcal{N}_c$ . The output of the last neural network is the cost function  $\mathcal{C}$ , that is  $\mathcal{C} = \mathcal{N}_c(\alpha, v, v')$ . The last neural network  $\mathcal{N}_c$  is assumed to be trained during the asset calibration, offline, and then, it is frozen, assumed to be known and accurate enough.

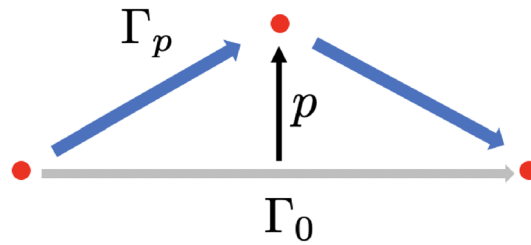
At this point, the EL equation can be enforced by computing the cost function derivatives  $\frac{\partial \mathcal{C}}{\partial v}$  and  $\frac{d}{ds} \left( \frac{\partial \mathcal{C}}{\partial v'} \right)$  by automatic differentiation, and the EL equation is finally enforced in the loss function. The loss function minimization allows the calculation of the  $\mathcal{N}_v$  parameters, that is, the  $\mathcal{N}_v$  training. The whole architecture and workflow is sketched in Fig. 1.

### Optimal planning on parametrized trajectories

This section generalizes the procedure described in the previous one, by considering parametrized trajectories  $\Gamma_{\mathbf{p}}$ , where vector  $\mathbf{p}$  groups the set of parameters involved in the description of the trajectory. Usual parametric descriptions are based on the use of polygonal curves or NURBS, whose parametric description involves the control points position, among many other possibilities.



**Fig. 1** Flow chart for the training of network  $\mathcal{NN}_v$ , when the cost function  $\mathcal{C}$  is approximated by a neural network based regression  $\mathcal{NN}_c$ . The weights and biases of  $\mathcal{NN}_v$  are noted  $\Theta$



**Fig. 2** Parametrized trajectory by using a single parameter  $p$  defining  $\Gamma_p$

A more implicit way of expressing possible trajectories consists on solving a parametric partial differential equation, in such a way that the level-set curves could represent valuable trajectories. This strategy was considered in our former work [16] for addressing deposition trajectories in additive manufacturing.

In what follows, for the sake of simplicity but without loss of generality, a simple case, involving a single parameter, noted by  $p$ , is considered, and illustrated in Fig. 2.

In the present case, the rationale described previously remains almost unchanged. Here, again, a cost function, analytical or expressed with the help of a DNN, is assumed to be available. This cost function is again represented by  $\mathcal{C}(s, v, v'; p)$ , where the dependence on  $p$  of the cost function is highlighted, even if in the sequel, this dependence will be omitted in the notation, for the sake of clarity, if there is no risk of confusion.

In this section the only difference is the domain in which  $s$  takes its values, which depends on the considered trajectory  $\Gamma_p$ . Each  $\Gamma_p$  involves a length  $S_p$ . Thus, the global cost reads:

$$\mathcal{I}_p = \int_0^{S_p} \mathcal{C}(s, v(s), v'(s)) ds. \tag{7}$$

**Optimal planning on each trajectory offline, and optimal trajectory extraction online**

The simplest procedure consists in considering a set of parameters in the interval of variation of  $p$ ,  $p \in [0, P]$ . When considering  $N$  values uniformly distributed in the interval  $[0, P]$ , they read:

$$p_i = (i - 1) \times \frac{P}{N - 1}, \quad i = 1, \dots, N. \tag{8}$$

For each trajectory  $\Gamma_{p_i}$ , the optimal velocity is computed following the rationale just described in “Optimality along a given trajectory”, which leads to  $v^{opt}(s \in \Gamma_{p_i})$ . Later on,

by introducing that optimal velocity into the total cost expression  $\mathcal{I}_p$ , it results in:

$$\mathcal{I}_{p_i} = \int_0^{S_{p_i}} C(s, v^{\text{opt}}(s), v'^{\text{opt}}(s)) ds. \quad (9)$$

Finally the best trajectory, represented by  $p^{\text{opt}}$  is obtained by solving the minimization problem:

$$p^{\text{opt}} = \underset{p=p_1, \dots, p_N}{\text{argmin}} \{ \mathcal{I}_p \}, \quad (10)$$

### Parametric optimal planning and optimal trajectory extraction

A more appealing route consists in using a reference domain where the curvilinear coordinate  $s$  is mapped into a reference interval,  $\xi \in [0, 1]$ . In the case considered here, illustrated in Fig. 2, the coordinate mapping reads;

$$\xi = \frac{s}{S(p)}, \quad (11)$$

with  $S(p) = 2 \sqrt{0.5^2 + p^2}$  when considering a unit distance between the departure and arrival points or, equivalently:

$$s = \xi S(p), \quad (12)$$

representing the direct and inverse mappings,  $\xi(s; p)$  and  $s(\xi; p)$ . Expressing the global cost in the reference domain

$$\mathcal{I}_\xi = \int_0^1 C(\xi, v(\xi), v'(\xi)) \frac{ds}{d\xi} d\xi, \quad (13)$$

with  $\frac{dv}{ds} = \frac{dv}{d\xi} \frac{d\xi}{ds} = v'(\xi) \frac{d\xi}{ds}$ , the optimal velocity results from the solution of the Euler–Lagrange equation.

### Discussion

In Appendix , several trajectory optimization problems associated to different cost functions are considered, solved and discussed, to emphasize the effects of the different terms involved in the cost function on the optimal trajectory, that as expected, in certain circumstances deviates from the shortest one.

The addressed problems do not have an intrinsic interest beyond the fact of illustrating how the different factors impacting the cost are addressed in the proposed formalism.

Even if the performed analyses proved that the proposed methodology is general enough for considering a diversity of scenarios of practical interest, its main limitation, as in many other engineering areas, is the parametrization itself.

Complex trajectories, as the ones concerned by drones operating in urban areas involving numerous non-flying zones (buildings, restricted areas, regions with intense wind turbulences or without satellite coverage,...) need rich enough parametrization able to represent trajectories in such multi-constraint domains. Thus, tens of parameters are needed to parametrize trajectories able to adapt to all the existing constraints. In that case, the procedures just described fail because of the curse of dimensionality.

The curse of dimensionality can be circumvented, or at least alleviated, by considering separated representations (tensor decompositions) of the multi-parametric space. Separated representations, at the heart of the so-called PGD—Proper Generalized Decomposition—, express multi-valued functions from a finite sum of products of lower

dimensional functions. These separated representations can be generated by projection (intrusive formulation) [11,12], or directly from the solutions related to a sampling of the high-dimensional parametric space (non-intrusive formulation), giving rise to multi-parametric surrogates [13,14].

The separated constructor employing projection (intrusive) allows accurate (and certifiable eventually) solutions, however, its deployment for addressing nonlinear or non-affine models remains a tricky issue.

On the other hand, the construction of multi-parametric surrogates is always possible, by combining the separated representation of the solutions, with appropriate regularizations (e.g., elastic-net, ridge, lasso,...), for pushing aside overfitting while promoting sparsity, looking for approximating complex nonlinear parametric solution from a very reduced amount of data (the numerical solutions related to the parameters sampling).

Even if the just referred parametric surrogates constitute an appealing route, the accuracy of the solution for a given choice of the parameters remains difficult to quantify, and with it, the possibility of certifying the derived decisions.

Due to the limitations of operating in parametric settings, non-parametric approaches seem preferable. Among the numerous alternatives, two procedures attracted our attention. The first consists of using reinforced learning—RL—, while the second consists of extending the Euler–Lagrange formalism to the construction of the optimal trajectory and the optimal operation along that optimal trajectory [7], while expressing the cost from a NN-based regression.

Even if RL is a very valuable approach, widely considered in a diversity of applications eluding a frugal parametric formulation, while addressing eventual variability, as it is the case of driver assistant, optimal planning, and more generally optimal decision making in complex environments, the training cost and complexity, as well as the prediction performances, make difficult, at present, its use in critical applications.

For that reason, in what follows, the last approach, the one based on the use of the standard Euler–Lagrange formalism operating on a NN-based cost function will be retained. This approach represents an appealing compromise between: (i) the explicability that the Euler–Lagrange formalism provides; (ii) the accuracy ensured by the offline construction of the cost function (calibration); and (iii) the training performances because no data is *a priori* required for evaluating the optimal solution.

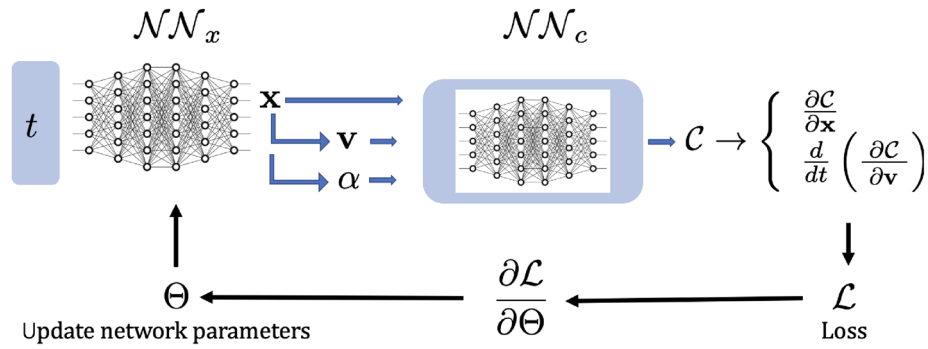
Thus, the resulting technique becomes a strong minimizer operating on an offline learnt cost function (calibration performed only once), the last employing experimental data, while the former (the minimizer) only consider the physical knowledge provided by the Euler–Lagrange equations.

This last approach will be considered in the next section, and then, its extension to stochastic settings accounting for the environment variability (time and space fluctuations) will be then put in perspective.

### **Optimal trajectory and planning**

By ignoring inertia, and referring the time by  $t$ , the most general setting consists in computing the optimal trajectory  $\mathbf{x}(t)$  and the optimal velocity along it,  $\mathbf{v}(t)$ , by applying the EL equation on the cost function.





**Fig. 3** Flow chart for the training of network  $\mathcal{NN}_x$ , when the cost function  $\mathcal{C}$  is approximated by a neural network based regression  $\mathcal{NN}_c$ . The weights and biases of  $\mathcal{NN}_x$  are noted  $\Theta$

The EL equation can be generalized to address higher-order derivatives. However, for the sake of simplicity and without loss of generality, in what follows we assume  $\mathcal{C}(t, \mathbf{x}, \mathbf{v})$ , with  $\mathbf{v} = \dot{\mathbf{x}}$ .

We consider the simple 2D cost function  $\mathcal{C} = (v_x + y)^2 + (v_y - x)^2$ , with  $(v_x, v_y)$  the components of the velocity vector  $\mathbf{v} = (\dot{x}, \dot{y})^T$ , and  $x, y$  the components of the position vector  $\mathbf{x} = (x, y)^T$ .

We assume that the departure and arrival points are known, as well as the time at which the arrival point is reached,  $t = T$ .

In the present case the EL equations read

$$\begin{cases} \frac{\partial \mathcal{C}}{\partial x} - \frac{d}{dt} \frac{\partial \mathcal{C}}{\partial \dot{x}} = 0, \\ \frac{\partial \mathcal{C}}{\partial y} - \frac{d}{dt} \frac{\partial \mathcal{C}}{\partial \dot{y}} = 0. \end{cases} \quad (14)$$

Considering the expression of  $\mathcal{C}$ , Eq. (14) leads to:

$$\begin{cases} \ddot{x} - x + 2\dot{y} = 0, \\ \ddot{y} - y - 2\dot{x} = 0. \end{cases} \quad (15)$$

The departing and arrival points are defined respectively by  $(0, 0)$  at  $t = 0$  and  $(0, H)$  at  $t = T$ .

The EL equations given in Eq. (15) can be solved by applying the finite difference method for example. The converged finite difference solution is considered as the reference solution in this section.

When the cost function is not available analytically, an appealing neural architecture is presented in Fig. 3. The fact of using two different neural networks, each predicting one component of the position vector  $\mathbf{x}$ , that is,  $x$  and  $y$ , provides better accuracy than using a single NN for predicting both components.

The solution of the problem for  $H = 1m$  and  $T = 1s$  is shown in Fig. 4, where the solutions obtained with three different strategies are compared with the reference one. The four strategies include:

- *Finite difference discretization.* The finite difference method is applied to solve monolithically both equations in (15), with a time discretization of  $n_t = 1000$  nodes uniformly distributed. The same time nodal distribution is considered in the different strategies described below.
- *Optimized cost.* The cost function  $\mathcal{C}$  is approximated by a neural network  $\mathcal{NN}_c$ , then the parameters of  $\mathcal{NN}_c$  are frozen. Later on,  $\mathcal{NN}_x$  is trained with a loss function  $\mathcal{L}_c$



**Table 1** The neural network used as the surrogate model

$\mathcal{NN}_x$		
Layer	Building block for $x$ prediction	Activation
1	Input layer having the size of the time $t$ coordinate (= 1)	No activation
2	Dense layer with 200 neurons	$\tanh$
3	Dense layer with 200 neurons	$\tanh$
4	Output layer having the size of $x$ per node (= 1)	$linear$
Independent networks are used for predicting the coordinates $x$ and $y$ , while having the same structure. $\tanh$ stands for the hyperbolic tangent		
Layer	Building block for $y$ prediction	Activation
1	Input layer having the size of the time $t$ coordinate (= 1)	No activation
2	Dense layer with 200 neurons	$\tanh$
3	Dense layer with 200 neurons	$\tanh$
4	Output layer having the size of $y$ per node (= 1)	$linear$

representing the total cost:

$$\mathcal{L}_c = \sum_{i=1}^{n_t} \mathcal{C}_i. \quad (16)$$

- *Lagrange Net*. This strategy uses the algorithm illustrated in Fig. 1. The cost function  $\mathcal{C}$  is approximated by the same neural network  $\mathcal{NN}_c$ , then the parameters of  $\mathcal{NN}_c$  are set to non-trainable. Later on,  $\mathcal{NN}_x$ , is trained with a loss function  $\mathcal{L}_{LN}$  defined by:

$$\mathcal{L}_{LN} = \int_0^T \left( \frac{\partial \mathcal{C}}{\partial x} - \frac{d}{dt} \frac{\partial \mathcal{C}}{\partial \dot{x}} \right)^2 + \left( \frac{\partial \mathcal{C}}{\partial y} - \frac{d}{dt} \frac{\partial \mathcal{C}}{\partial \dot{y}} \right)^2 dt. \quad (17)$$

- *Physics Informed Neural Network, PINN*. The PINN is employed to solve Eq. (15). The same neural network architecture used for  $\mathcal{NN}_x$  is used again for the PINN, with the same boundary conditions and same time mesh. The PINN loss function,  $\mathcal{L}_{PINN}$ , reads:

$$\mathcal{L}_{PINN} = \int_0^T ((\ddot{x} - x + 2\dot{y})^2 + (\ddot{y} - y - 2\dot{x})^2) dt \quad (18)$$

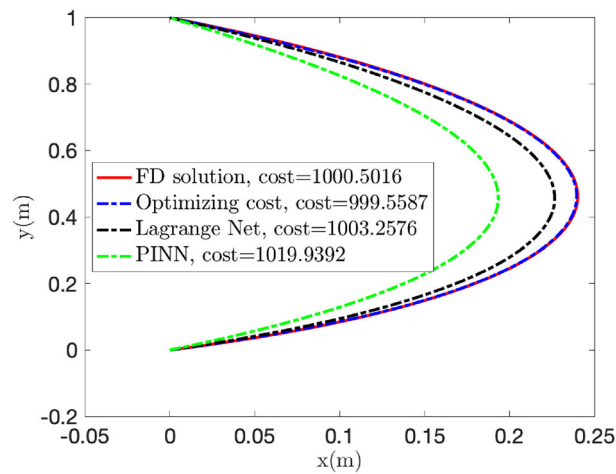
One may note that the PINN consider the explicit expression of the cost function  $\mathcal{C}$  instead of the neural network  $\mathcal{NN}_c$  that was considered by both previous strategies.

The same training methodology is used for the three neural networks  $\mathcal{NN}_x$ : PINN, Lagrange Net and Optimized Cost.

It is important to mention that the considered architectures do not need a data-based training. Here, the loss is based on the deviation from the existing knowledge, the optimality conditions in our case. In a similar way than PINNs, here we are closer from a solver than from a learner. The issue related to the choice of the networks hyper-parameters is the usual one, where a certain number of trials are made until obtaining performances considered good enough.

### A first training

The ADAM stochastic gradient descent algorithm is considered, with a learning rate of  $10^{-4}$  and 2000 epochs. The  $\mathcal{NN}_x$  neural network structure is illustrated in Table 1.



**Fig. 4** Solution for  $C = (v_x + y)^2 + (v_y - x)^2$ ,  $H = 1\text{ m}$  and  $T = 1\text{ s}$  when using finite difference (FD), optimizing the cost directly, the Lagrange-Net flow chart illustrated in Fig. 3, and a Physics Informed Neural Network (PINN) solving the equations illustrated in Eq. (15)

**Table 2** The deeper neural network used as the surrogate model  $\mathcal{NN}_x$

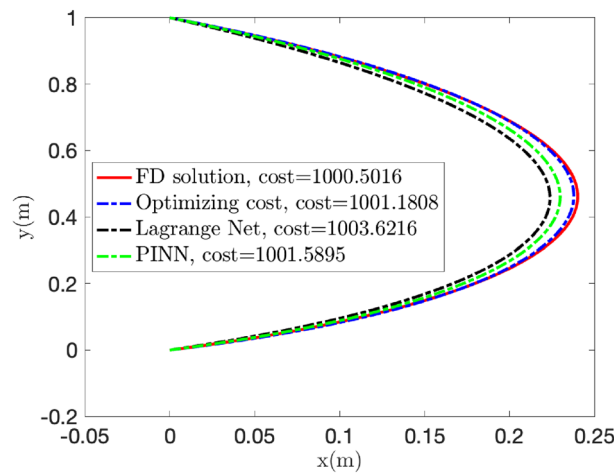
Layer	Building block for $x$ prediction	Activation
1	Input layer having the size of the time $t$ coordinate ( $= 1$ )	No activation
2	Dense layer with 60 neurons	<i>relu</i>
3	Dense layer with 60 neurons	<i>relu</i>
4	Dense layer with 60 neurons	<i>relu</i>
5	Dense layer with 60 neurons	<i>relu</i>
6	Output layer having the size of $x$ per node ( $= 1$ )	<i>linear</i>
Layer	Building block for $y$ prediction	Activation
1	Input layer having the size of the time $t$ coordinate ( $= 1$ )	No activation
2	Dense layer with 60 neurons	<i>relu</i>
3	Dense layer with 60 neurons	<i>relu</i>
4	Dense layer with 60 neurons	<i>relu</i>
5	Dense layer with 60 neurons	<i>relu</i>
6	Output layer having the size of $y$ per node ( $= 1$ )	<i>linear</i>

Independent networks are used for predicting the coordinates  $x$  and  $y$ , while having the same structure. *relu* stands for the rectified linear unit

Figure 4 compares the computed trajectories  $\mathbf{x}(t)$ , from which we can conclude that techniques involving derivatives calculation exhibit lower accuracy. The indicated cost in Fig. 4 is computed as the sum of the cost  $C$  over all the time steps, as expressed in Eq. (16). The fact that the optimized cost is slightly smaller than the reference one can be explained by the fact that the former uses an approximation of the cost, that is, it proceeds from  $\mathcal{NN}_C$ .

### Training deeper networks

The used training algorithm is again ADAM, with a learning rate of  $10^{-3}$  and 100 epochs of gradient descent. The  $\mathcal{NN}_x$  neural network structure used in this section is illustrated in Table 2. The optimal trajectories using the different schemas are illustrated in Fig. 5.



**Fig. 5** Trajectory solutions when using the deeper network illustrated in table 2

**Table 3** A deeper neural network used as the surrogate model  $\mathcal{NN}_x$  combining *relu* and *tanh* activation functions

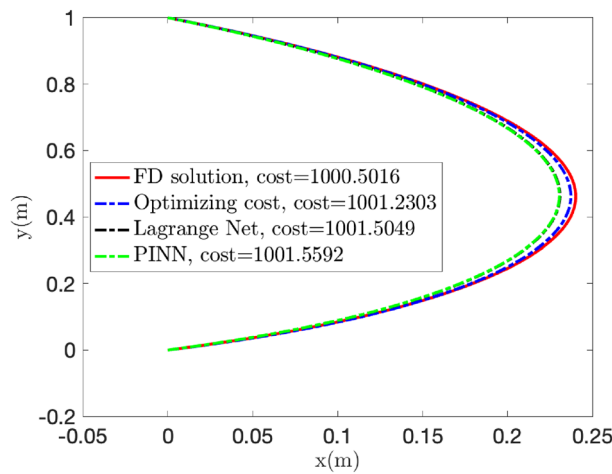
Layer	Building block for $x$ prediction	Activation
1	Input layer having the size of the time $t$ coordinate (= 1)	No activation
2	Dense layer with 200 neurons	<i>tanh</i>
3	Dense layer with 60 neurons	<i>relu</i>
4	Dense layer with 60 neurons	<i>relu</i>
5	Dense layer with 60 neurons	<i>relu</i>
6	Output layer having the size of $x$ per node (= 1)	<i>linear</i>
Layer	Building block for $y$ prediction	Activation
1	Input layer having the size of the time $t$ coordinate (= 1)	No activation
2	Dense layer with 200 neurons	<i>tanh</i>
3	Dense layer with 60 neurons	<i>relu</i>
4	Dense layer with 60 neurons	<i>relu</i>
5	Dense layer with 60 neurons	<i>relu</i>
6	Output layer having the size of $y$ per node (= 1)	<i>linear</i>

It can be noticed that a deeper network with a rectified linear unit (*relu*) activation does not produce a significant improvement in the Lagrange Net results, even if it improves the PINN performances.

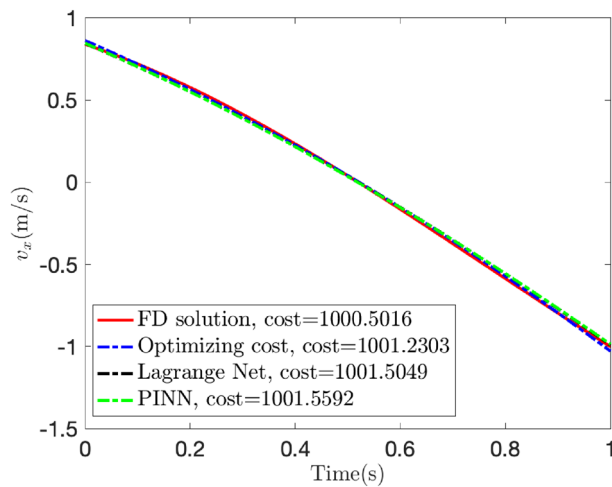
Thus, a third model, combining ReLU and tanh activations, is trained using the structure given in Table 3, that improves the Lagrange Net results, as Fig. 6 proves. The associated velocities all along the trajectory, are compared for the horizontal and vertical velocity components in Figs. 7 and 8, respectively.

The results obtained with the three strategies exhibit similar accuracy. Their main difference concerns the expression of the cost function, explicit in the case of the PINN approach, and expressed from a general regression in the other two cases.

It is important to mention that the main aim of the previous analysis was proving that different regressions, making use of a multi-layer perceptron trained from different loss functions, can be successfully used, reaching equivalent accuracies. The previous analysis is not expected providing a comparison among them. Such a comparison is beyond the objectives of the present paper.



**Fig. 6** Trajectory solutions when using the deeper network illustrated in Table 3, combining *relu* and *tanh* activation functions



**Fig. 7** Optimal velocities  $v_x$  when using the deeper network illustrated in Table 3, combining *relu* and *tanh* activation functions

**Future prospects: operating in stochastic environments**

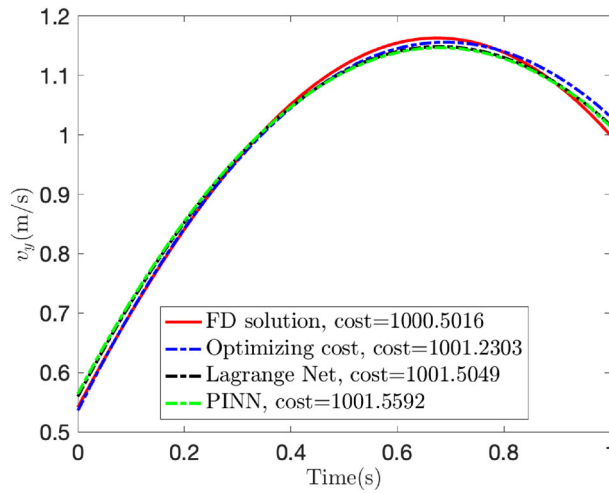
This section aims at determining the optimal velocity when the cost has a given statistical variability. For that purpose we assume the flight of a drone, where the resistance scales with the drone-wind relative velocity.

For the sake of simplicity we assume a rectilinear drone trajectory, with the wind (uniform in space) aligned with the drone trajectory and characterized by a certain statistical distribution. One expects that during the drone flight the wind velocity will represent the entire known statistics (ergodicity assumption).

The considered cost function consists of two terms (here the inertia term is neglected). The first representing the drag  $C_d$ , scaling with

$$C_d \sim (v_d - v)^2 v_d, \tag{19}$$

with  $v_d$  the drone velocity along the rectilinear trajectory and  $v$  the one of the wind, uniform in space but that evolves in time.



**Fig. 8** Optimal velocities  $v_y$  when using the deeper network illustrated in Table 3, combining *relu* and *tanh* activation functions

The second contribution to the cost concerns the flight duration. Thus, if the path, of length  $L$ , is expected be completed in the time period  $T$ , the nominal velocity will result  $v_d^{nom} = L/T$ , and then, any deviation from it, is considered as an extra-cost. By assuming a constant velocity of the drone, the cost could be expressed from

$$C_t \sim (v_d T - L)^2. \tag{20}$$

Both contributions are combined into the global cost function

$$C = C_d + C_t = (v_d - v)^2 v_d + \lambda (v_d T - L)^2, \tag{21}$$

with  $\lambda$  a coefficient weighting both contributions.

Now, being the wind velocity uniform in space, as well as the one of the drone, the global minimization coincides with the local one. The later reads:

$$\frac{\partial C}{\partial v_d} = 0, \tag{22}$$

leading to

$$3v_d^2 - (4v - 2\lambda T^2)v_d + v^2 - 2\lambda TL = 0, \tag{23}$$

that, in absence of traveling time constraints, that is, with  $\lambda = 0$ , the minimum cost concerns, as expected,  $v_d = v$ .

Now, we assume that the wind velocity is not constant in time, and that during the flight, the wind velocity will represent the entire wind statistics.

Thus, for any possible wind velocity  $v$ , its probability (fraction of the flight time in which the wind has that velocity) is given by  $p(v)$ . The optimal drone velocity when the wind velocity is  $v$ , is given by the solution of Eq. (23), whose associated cost  $C(v)$  results from applying Eq. (21).

If, for example, we assume a uniform wind velocity distribution in the interval  $v \in [v_{min}, v_{max}]$ ,  $p(v) = 1/l$ , with  $l = v_{max} - v_{min}$ , then, it is enough to consider the mean of the cost, that is

$$\bar{C} = \int_{v_{min}}^{v_{max}} C \frac{1}{l} dv, \tag{24}$$

then, compute the wind velocity  $v^*$  related to  $\bar{C}$ , i.e.  $C(v^*) = \bar{C}$ , and finally the drone velocity associated to  $v^*$  according to Eq. (21), that represents the optimal constant drone velocity.

For the sake of illustrative purposes we consider the following numerical values (in the metric system):  $T = 10$ ,  $L = 50$ ,  $v_{\max} = 19$ ,  $v_{\min} = 1$  and  $\lambda = 0.06$ . In that case the optimal drone velocity that the previous procedure provides becomes  $v_d = 9.13$ , whereas the optimal drone velocity related to the mean wind velocity  $\bar{v}_w = 0.5(v_{\max} + v_{\min})$  is  $v_d(\bar{v}_w) = 7.5$ .

Thus, the optimal drone velocity becomes different from the one related to the wind velocity mean value. This is due to the fact that the cost is a nonlinear function.

The proposed procedure can of course be generalized for more complex scenarios. For that purpose, the just described methodology should be deployed into the two frameworks described in the present paper, the one concerning the optimal velocity on a given trajectory, and the other where both the optimal trajectory and the optimal velocity on it are computed simultaneously.

The just proposed rationale can be embedded into a neural architecture that using the wind velocity as an input computes the optimal drone velocity, in such a way that the associated cost is minimized. Then, by considering a variability around the wind velocity, the optimal drone velocity is obtained by enforcing that the resulting average cost is minimized.

Therefore, both procedures considered in the present paper, apply by replacing the cost by its average. The implementation, analysis of performances and applications, constitute the topic of a paper in progress.

## Conclusion

This work proposes a holistic approach to trajectory and velocity optimization along a path, while minimizing the cost based on the Euler–Lagrange functional minimization, along with the use of artificial neural networks for machine learning.

Three different scenarios were addressed: (i) the one in which the trajectory is done, and the velocity on it must be obtained for minimizing the cost; (ii) the same procedure but now operating on parametrized trajectories, that is, trajectories defined by a number of parameters, as it is the case when describing trajectories from splines; and finally, (iii) when the trajectory and the velocity along it are computed simultaneously from the cost minimization. The three cases were formulated and their solutions analyzed. Moreover, the effect of considering variability was discussed and put in perspective.

Moreover, two approaches were considered, the one in which the cost function is explicitly defined, and the one in which the cost function is expressed from a NN-based approximation, the later offering higher versatility and generality.

## Acknowledgements

This research is part of the DesCartes programme and is supported by the National Research Foundation, Prime Minister Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. This project has received funding from the European Union Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 956401 (XS-Meta). This work has been partially funded by the Spanish Ministry of Science and Innovation, AEI /10.13039/501100011033, through Grant number PID2020-113463RB-C31 and the Regional Government of Aragon and the European Social Fund, group T24-20R. The authors also acknowledge the support of ESI Group through the chairs at the University of Zaragoza and at ENSAM Institute of Technology and the support of the SKF Group research chair at Arts et Metiers Institute of Technology.

**Author contributions**

All the authors participated equally in the paper content.

**Availability of data and materials**

Data will be made available upon request.

**Declarations****Competing interests**

The authors declare no competing interests.

**Appendix A: Numerical experiments and discussion****A.1 Cost function**

In this numerical example, the following cost function is considered

$$C = \frac{1}{2} (v^2(s) + \kappa p v^2(s) + \beta(v(s) - \bar{v})^2 + \gamma v'^2), \quad (25)$$

where  $\kappa p v^2(s)$  represents a trajectory-dependent cost, and the parameter  $\gamma$  is introduced to easily evaluate solutions when the cost associated to  $v'(s)$  is neglected.

**A.2 Euler–Lagrange equation**

For the cost function given in Eq. (25), the derivative terms involved in the EL equation write:

$$\frac{\partial C}{\partial v} = v + \kappa p v + \beta(v - \bar{v}), \quad (26)$$

and

$$\frac{d}{ds} \left( \frac{\partial C}{\partial v'} \right) = \gamma v'', \quad (27)$$

from which, the EL equation gives:

$$\gamma v''(s) - (1 + \kappa p + \beta)v(s) = -\beta\bar{v}. \quad (28)$$

The imposed boundary conditions for this problem are  $v(s = 0) = v(s = S(p)) = 0$ .

**A.3 Expressing the Euler–Lagrange equation in the reference domain**

By considering the mapping introduced in Eqs. (11) and (12), one can write:

$$\frac{\partial}{\partial s} = \frac{\partial \xi}{\partial s} \frac{\partial}{\partial \xi} = \frac{1}{\sqrt{1 + 4p^2}} \frac{\partial}{\partial \xi}, \quad (29)$$

and

$$\frac{\partial^2}{\partial s^2} = \frac{1}{1 + 4p^2} \frac{\partial^2}{\partial \xi^2}. \quad (30)$$

Equations (29) and (30) allow us to express the EL equation in the reference domain:

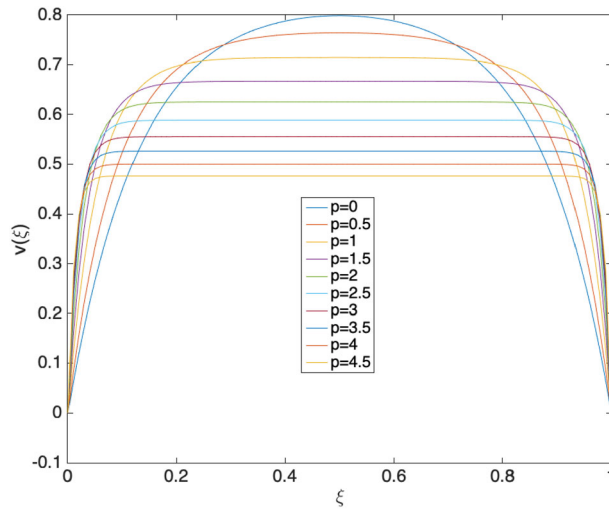
$$\frac{\gamma}{1 + 4p^2} v''(\xi) - (1 + \kappa p + \beta)v(\xi) = -\beta\bar{v}. \quad (31)$$

**A.4 Inertia-less limit**

When the inertia can be neglected in the cost function, i.e.,  $\gamma = 0$ , the ODE equation (31) reduces to:

$$(1 + \kappa p + \beta)v(\xi) = \beta\bar{v}, \quad (32)$$





**Fig. 9** Comparing the optimal velocity on different trajectories

which allows defining the optimal velocity as:

$$v(\xi) \equiv v^{opt}(\xi) = \frac{\beta}{1 + \kappa p + \beta} \bar{v}. \tag{33}$$

One can note that the optimal velocity decreases with increasing  $\kappa$  and  $p$ , and approaches  $\bar{v}$  when  $\beta \rightarrow \infty$ .

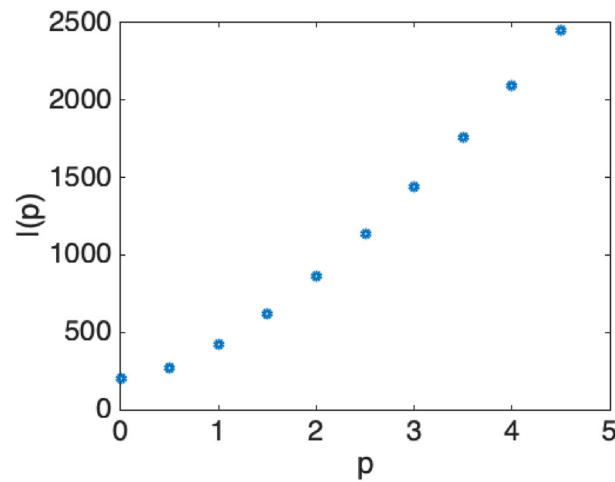
**A.5 Numerical solution for increasing  $p$**

By considering  $\kappa = 1$ ,  $\beta = 5$ ,  $\gamma = 0.1$  and  $\bar{v} = 1$ , Fig. 9 shows the optimal velocity profiles for  $p = 0, 0.5, 1, 1.5, \dots, 4.5$ . By comparing the solutions for  $p = 0$  and  $p = 5$ , it can be noticed that the higher is the value of  $p$ , the more intense is the resistance to movement (higher cost) and, consequently, the maximum velocity decreases. The total costs when considering the just computed optimal velocities result:  $\mathcal{I}(p = 0) = 197.7$  and  $\mathcal{I}(p = 5) = 2818.6$ , which proves that the rectilinear trajectory becomes the optimal one. In fact, the resistance (cost induced by trajectories deviating from the rectilinear one defined by  $p = 0$ ) increases by increasing  $p$ . Figure 10 shows the associated total cost associated to the optimal velocity profiles.

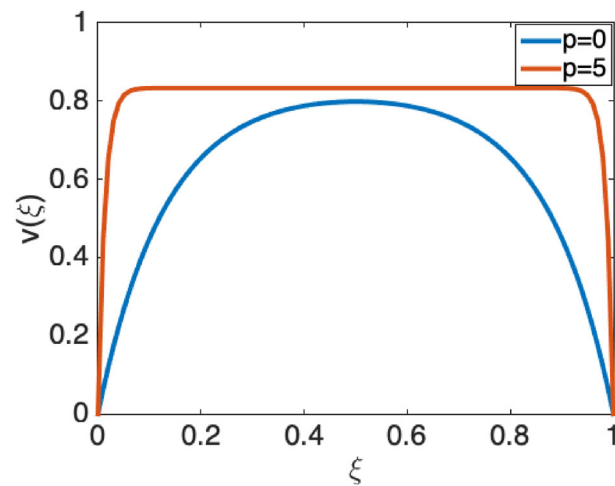
**A.6 Evaluating effect of trajectory length**

In this section, to evaluate the impact of the trajectory length on the planning, we remove the dependence of the cost function on the parameter  $p$  by setting  $\kappa = 0$ . Thus, from the point of view of the cost function  $\mathcal{C}$ , no difference between one trajectory or another exists. However, from the point of view of the global cost  $\mathcal{I}$ , the longer is the trajectory, the higher is the cost. Consequently, one expects that, again, the rectilinear trajectory ( $p = 0$ ) is the optimal one, whereas the longest implies higher cost because of the necessity of covering a longer distance.

Figure 11 compares the optimal velocity profiles  $v^{opt}(\xi; p)$ , for  $p = 0$  and  $p = 5$  in absence of a cost term depending on  $p$ . As expected, when computing the global cost related to the optimal velocities profiles, the longer trajectory ( $p = 5$ ) presents the higher cost:  $\mathcal{I}(p = 0) = 197.7$  and  $\mathcal{I}(p = 5) = 1009.7$ .



**Fig. 10** Cost evolution  $\mathcal{I}(p)$



**Fig. 11** Comparing the optimal velocity on two trajectories characterized by  $p = 0$  and  $p = 5$  respectively, when  $\kappa = 0$

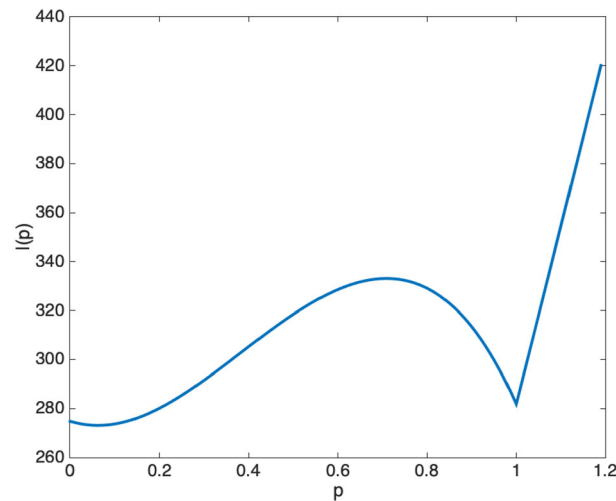
**A.7 Optimal trajectories differing from the rectilinear one**

In this section we set  $\kappa = 5$ , but a vanishing resistance for a trajectory that deviates from the rectilinear one. For example, this can be achieved by setting  $p = 1$ , with  $\beta = 4, \gamma = 0.1$  and  $\bar{v} = 1$ , and the following cost function:

$$\mathcal{C} = \frac{1}{2} \left( v^2(s) + \kappa \sqrt{(p - 1)^2 v^2(s) + \beta(v(s) - \bar{v})^2 + \gamma v'^2} \right). \tag{34}$$

The just introduced cost function enforces the fact that when  $p = 1$  and the trajectory deviates a bit from the rectilinear one, the trajectory-dependent cost, due to the effect of other factors (environmental conditions for example) vanishes. Such a choice allows enforcing the fact that the optimal trajectory could be different from the rectilinear one.

Figure 12 shows the evolution of the total cost  $\mathcal{I}(p)$  evaluated by considering the optimal velocities on each trajectory characterized by the parameter  $p$ . Figure 12 shows that the optimal trajectory is located somewhere between the rectilinear shortest path (but having a high cost induced by the deviation from  $p = 1$ ), and the one in which the contribution



**Fig. 12** Cost evolution  $\mathcal{J}(p)$  associated with the cost function depicted in Eq. (34)

scaling with  $|p - 1|$  is minimum (however exhibiting a longer path), with the minimum global cost for a trajectory close to  $p = 0.1$  in the present example.

*Remark.* In all the discussion until now, the cost function involves a term penalizing deviations with respect to a reference velocity  $\bar{v}$ . However, other forms of this term can be designed to penalize the deviation with respect to the time spent to cover the whole trajectory for example. The travel time can be expressed from  $\int_0^S \frac{ds}{v(s)}$ , and if we are looking for a duration  $T$ , the local cost could be expressed by  $\left(\frac{1}{v} - \frac{T}{S}\right)$ , that integrated from  $s = 0$  to  $s = S$  vanishes when the travel time is  $T$ , and becomes positive if the duration is longer and negative if it is shorter.

Received: 13 December 2023 Accepted: 9 April 2024  
Published online: 29 April 2024

## References

- Bae S, Kim Y, Guanetti J, Borrelli F, Moura S. Design and implementation of ecological adaptive cruise control for autonomous driving with communication to traffic lights. In: 2019 American Control Conference (ACC), pp. 4628–4634, 2019. <https://doi.org/10.23919/ACC.2019.8814905>.
- Guanetti J, Kim Y, Borrelli F. Control of connected and automated vehicles: state of the art and future challenges. *Ann Rev Control.* 2018;45:18–40. <https://doi.org/10.1016/j.arcontrol.2018.04.011>.
- Lipp T, Boyd SP. Minimum-time speed optimisation over a fixed path. *Int J Control.* 2014;87:1297–311.
- De Filippis G, Lenzo B, Sorniotti A, Sannen K, De Smet J, Gruber P. On the energy efficiency of electric vehicles with multiple motors. In: 2016 IEEE Vehicle Power and Propulsion Conference (VPPC), pp. 1–6, 2016. <https://doi.org/10.1109/VPPC.2016.7791737>.
- Sforza A, Lenzo B, Timponi F. A state-of-the-art review on torque distribution strategies aimed at enhancing energy efficiency for fully electric vehicles with independently actuated drivetrains. *Int J Mech Control.* 2019;20:3–15.
- Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning. *Int J Robot Res.* 2011;30(7):846–94. <https://doi.org/10.1177/0278364911406761>.
- Sachenbacher M, Leucker M, Artmeier A, Haselmayr J. Efficient energy-optimal routing for electric vehicles. In: AAAI, pp. 1402–1407, 2011.
- Sutton RS, Barto AG. Reinforced learning. An introduction. The MIT Press.
- Kim H, Pyeon H, Park JS, Hwang JY, Lim S. Autonomous vehicle fuel economy optimization with deep reinforcement learning. *Electronics.* 2020; 9(11). <https://doi.org/10.3390/electronics9111911>
- Ghnatios C, di Lorenzo D, Champaney V, Cueto E, Chinesta F. Optimal velocity planning based on the solution of the Euler–Lagrange equations with a neural network based velocity regression. *Discrete and Continuous Dynamical Systems.* 2023. <https://doi.org/10.3934/dcds.2023080>.
- Chinesta F, Huerta A, Rozza G, Willcox K. Model order reduction. In: *The Encyclopedia of Computational Mechanics.* 2nd Edn. 2015.
- Chinesta F, Cueto E, Abisset-Chavanne E, Duval JL, Khaldi FE. Virtual, digital and hybrid twins: a new paradigm in data-based engineering and engineered data. *Arch Comput Methods Eng.* 2020;27:105–34.

13. Sancarlos A, Champany V, Cueto E, Chinesta F. Regularized regressions for parametric models based on separated representations. *Adv Model Simul Eng Sci.* 2023;4:10.
14. Chinesta F, Cueto E. Empowering engineering with data, machine learning and artificial intelligence: a short introductory review. *Adv Model Simul Eng Sci.* 2023;9:21.
15. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys.* 2019;378:686–707. <https://doi.org/10.1016/j.jcp.2018.10.045>.
16. Quaranta G, Haug E, Duval JL, Chinesta F. Parametric evaluation of part distortion in additive manufacturing processes. *Int J Mater Form.* 2020;13:29–41.

### **Publisher's Note**

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.