# Optimal trajectory planning combiningmodel-based and data-driven hybrid approaches

**Chady Ghnatios**
ENSAM: Ecole Nationale Superieure d'Arts et Metiers
**Daniele di Lorenzo**
ENSAM: Ecole Nationale Superieure d'Arts et Metiers
**Victor Champaney**
ENSAM: Ecole Nationale Superieure d'Arts et Metiers
**Amine Ammar**
ENSAM: Ecole Nationale Superieure d'Arts et Metiers
**Cueto Elias** ( Ecueto@unizar.es )
Universidad de Zaragoza Escuela de Ingenieria y Arquitectura https://orcid.org/0000-0003-1017-4381
**Francisco Chinesta**
ENSAM: Ecole Nationale Superieure d'Arts et Metiers

**Research Article**

**Keywords:** Machine Learning, Artificial Intelligence, Optimal planning, Trajectory optimization, Euler-Lagrange equations, Physics informed machine learning

**Posted Date:** December 25th, 2023

**DOI:** https://doi.org/10.21203/rs.3.rs-3754087/v1

RESEARCH

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

Chady Ghnatios[1*], Daniele Di Lorenzo[2], Victor Champaney[2], Amine Ammar[3,4], Elias Cueto[5] and Francisco Chinesta[2,4]

*Correspondence:
Chady.Ghnatios@ensam.eu
[1] PIMM Lab & SKF Chair, Arts et Metiers Institute of Technology, Paris, France
Full list of author information is available at the end of the article

**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 a 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 (in the sense of minimizing a given cost function) 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

## 1 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, 4, 5, 6, 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 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, 12, 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, such parametric optimal trajectories framework faces the difficulty of addressing limitations imposed by black-box cost functions. For instance, when the cost is obtained by testing the asset under several operating scenarios, and then correlating the input features with the cost function, by using deep neural networks. In such a case, the Euler-Lagrange equation must be solved in an unusual manner, as detailed later in this work. Physics Informed Neural Networks (PINNs) offer an appealing gateway for performing on those 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 Section 2. The

second case, which tackles the case of parametric trajectories and the extraction of the optimal path through parameter optimization, is discussed in Section 3. Section 4 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, Section 5 discusses the route to extend the approach to stochastic settings.

## 2 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]. 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 $\mathtt{S}$ the total path length, the total cost associated with the trajectory $\Gamma$, $\mathcal{I}_\Gamma$, can be obtained using the integral

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

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

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

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 $v^{\mathtt{opt}}(s)$.

### 2.1 Cost functions expressed analytically

Let us consider, for example, the instantaneous cost function given by $\mathcal{C} = \frac{1}{2}v^2(s) + \frac{\beta}{2}(v(s) - \overline{v})^2 + \frac{1}{2}v'^2(s)$, where $\beta$ is the weight enforcing the target velocity $\overline{v}$. The optimal velocity results from the EL equation solution, which, in the present case, takes the form:

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

and

$$\frac{d}{ds}\frac{\partial \mathcal{C}}{\partial v'} = v'', \tag{4}$$

which result in the following ODE:

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

The integration of Eq. (5) requires two boundary conditions. In this example, we consider vanishing initial and final velocities, $v(s=0) = v(s=\mathtt{S}) = 0$. Solving the ODE problem with the appropriate boundary conditions results in the optimal velocity $v^{\mathtt{opt}}(s)$ 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 $\mathcal{C} = \frac{1}{2}v^2(s) + \frac{\beta}{2}(v(s) - \overline{v})^2$, and the optimal velocity can be trivially obtained from:

$$v(s) \equiv v^{\mathtt{opt}}(s) = \frac{\beta}{1+\beta}\overline{v}. \tag{6}$$

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

### 2.2 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 $v = \mathcal{NN}_v(s)$. Then, automatic differentiation allowed us to compute $v'(s)$. The velocity and its derivative, as well as the contribution to the cost function, depend on the potion $s$, here called $\alpha(s)$, are the inputs of the cost function neural network $\mathcal{NN}_c$. The output of the last neural network is the cost function $\mathcal{C}$, that is $\mathcal{C} = \mathcal{NN}_c(\alpha, v, v')$. The last neural network $\mathcal{NN}_c$ 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 $\mathcal{NN}_v$ parameters, that is, the $\mathcal{NN}_v$ training. The whole architecture and workflow is sketched in Fig. 1.

## 3 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.

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 Figure 2.

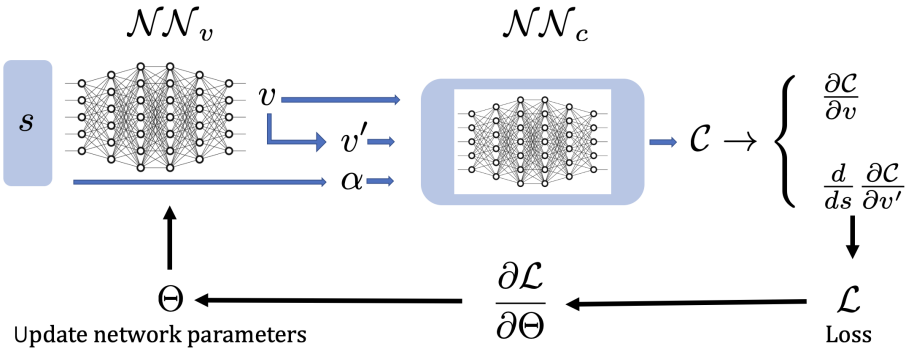


**Figure 1** Flow chart for the training of network $\mathcal{NN}_v$, when the cost function $\mathcal{C}$ is approximated by a neural network based regression $\mathcal{NN}_c$. The weights and biases of $\mathcal{NN}_v$ are noted $\Theta$.

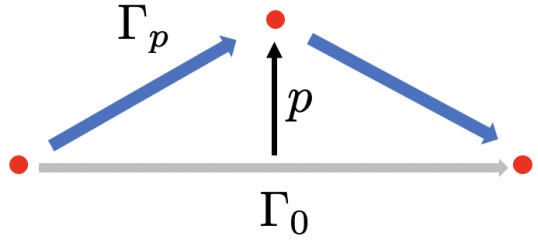


**Figure 2** Parametrized trajectory by using a single parameter $p$ defining $\Gamma_p$.

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 $\mathtt{S}_p$. Thus, the global cost reads:

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

### 3.1 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 $p$, $p \in [0, \mathtt{P}]$. When considering $\mathtt{N}$ values uniformly distributed in the

interval $[0, \mathtt{P}]$, they read:

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

For each trajectory $\Gamma_{p_i}$, the optimal velocity is computed following the rationale just described in Section 2, which leads to $v^{\mathtt{opt}}(s \in \Gamma_{p_i})$. Later on, by introducing that optimal velocity into the total cost expression $\mathcal{I}_p$, it results in:

$$\mathcal{I}_{p_i} = \int_0^{\mathtt{S}_{p_i}} \mathcal{C}(s, v^{\mathtt{opt}}(s), v'^{\,\mathtt{opt}}(s)\, ds. \tag{9}$$

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

$$p^{\mathtt{opt}} = \underset{p=p_1,\ldots,p_{\mathtt{N}}}{\mathtt{argmin}} \left\{\mathcal{I}_p\right\}, \tag{10}$$

### 3.2 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;

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

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

$$s = \xi\ \mathtt{S}(p), \tag{12}$$

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

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

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.

### 3.3 Numerical experiments and discussion

#### *3.3.1 Cost function*

In this numerical example, the following cost function is considered

$$\mathcal{C} = \frac{1}{2}\left(v^2(s) + \kappa p v^2(s) + \beta(v(s) - \overline{v})^2 + \gamma v'^2\right), \tag{14}$$

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.

### *3.3.2 Euler-Lagrange equation*

For the cost function given in Eq. (14), the derivative terms involved in the EL equation write:

$$\frac{\partial \mathcal{C}}{\partial v} = v + \kappa p v + \beta(v - \overline{v}), \tag{15}$$

and

$$\frac{d}{ds}\left(\frac{\partial \mathcal{C}}{\partial v'}\right) = v'', \tag{16}$$

from which, the EL equation gives:

$$\gamma v''(s) - (1 + \kappa p + \beta) v(s) = -\beta \overline{v}. \tag{17}$$

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

### *3.3.3 Expressing the Euler-Lagrange equation in the reference domain*

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

$$\frac{\partial}{\partial s} = \frac{\partial \xi}{\partial s}\frac{\partial}{\partial \xi} = \frac{1}{\sqrt{1 + 4p^2}}\frac{\partial}{\partial \xi}, \tag{18}$$

and

$$\frac{\partial^2}{\partial s^2} = \frac{1}{1 + 4p^2}\frac{\partial}{\partial \xi}. \tag{19}$$

Equations (18) and (19) allow us to express the EL equation in the reference domain:

$$\frac{\gamma}{1 + 4p^2} v''(\xi) - (1 + \kappa p + \beta) v(\xi) = -\beta \overline{v}. \tag{20}$$

### *3.3.4 Inertia-less limit*

When the inertia can be neglected in the cost function, i.e., $\gamma = 0$, the ODE equation (20) reduces to:

$$(1 + \kappa p + \beta) v(\xi) = \beta \overline{v}, \tag{21}$$

which allows defining the optimal velocity as:

$$v(\xi) \equiv v^{\mathtt{opt}}(\xi) = \frac{\beta}{1 + \kappa p + \beta}\overline{v}. \tag{22}$$

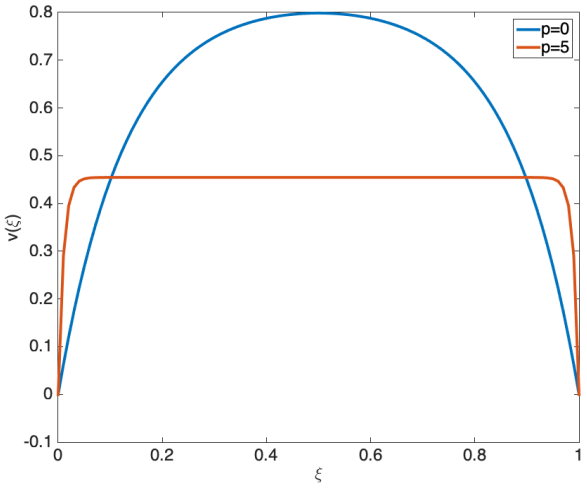


**Figure 3** Comparing the optimal velocity on two trajectories characterized by $p = 0$ and $p = 5$ respectively, with vanishing $v(\xi = 0) = 0$ and $v(\xi = 1) = 0$.

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

### *3.3.5 Numerical solution for increasing p*

By considering $\kappa = 1$, $\beta = 5$, $\gamma = 0.1$ and $\overline{v} = 1$, Fig. 3 compares the solutions for $p = 0$ and $p = 5$, from which 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 $p$.

To further illustrate the velocity profiles and the associate cost evolution with $p$, Fig. 4 shows the optimal velocity profiles for $p = 0, 0.5, 1, 1.5, \ldots, 4.5$ and Fig. 5 the associated total cost with the optimal velocity profiles.

### *3.3.6 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.

Figure 6 compares the optimal velocity profiles $v^{\mathtt{opt}}(\xi; p)$, for $p = 0$ and $p = 5$ in absence of a cost term depending on $p$. As expected, when computing the global

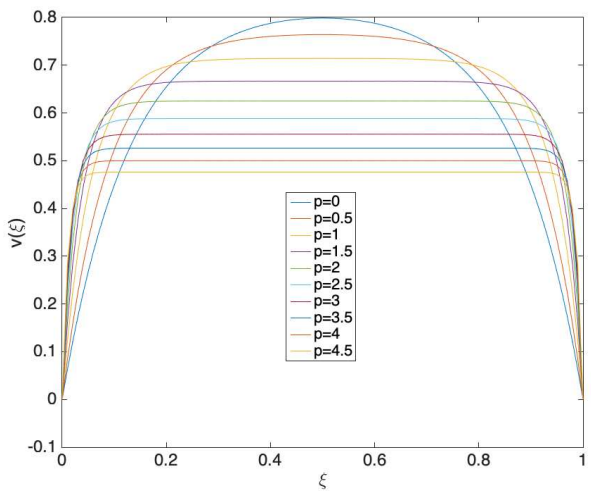


**Figure 4** Comparing the optimal velocity on different trajectories.

cost related to the optimal velocities profiles, the longer trajectory ($p = 5$) presents the higher cost: $\mathcal{I}(p = 0) = 197.7$ and $\mathcal{I}(p = 5) = 1009.7$.

### *3.3.7 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 $\overline{v} = 1$, and the following cost function:

$$\mathcal{C} = \frac{1}{2}\left(v^2(s) + \kappa\sqrt{(p-1)^2}v^2(s) + \beta(v(s) - \overline{v})^2 + \gamma v'^2\right), \tag{23}$$

Figure 7 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 7 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 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 $\overline{v}$. 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.

## 4 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.

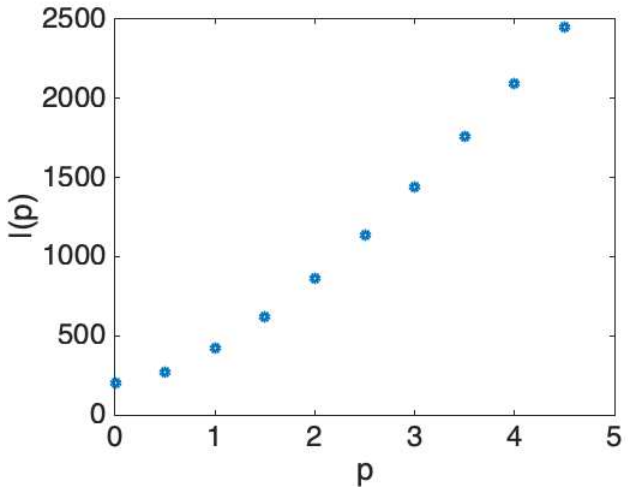


**Figure 5** Cost evolution $\mathcal{I}(p)$.

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

$$\begin{cases} \frac{\partial \mathcal{C}}{\partial x} - \frac{d}{dt}\frac{\partial \mathcal{C}}{\partial \dot{x}} = 0, \\ \frac{\partial \mathcal{C}}{\partial y} - \frac{d}{dt}\frac{\partial \mathcal{C}}{\partial \dot{y}} = 0. \end{cases} \tag{24}$$

Considering the expression of $\mathcal{C} = (v_x + y)^2 + (v_y - x)^2$, equation (24) leads to:

$$\begin{cases} \ddot{x} - x + 2\dot{y} = 0, \\ \ddot{y} - y - 2\dot{x} = 0. \end{cases} \tag{25}$$

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. (25) 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. 8.

The solution of the problem for $H = 1m$ and $T = 1s$ is shown in Fig. 9, where the solutions obtained with three different strategies are compared with the reference one. The four strategies include:

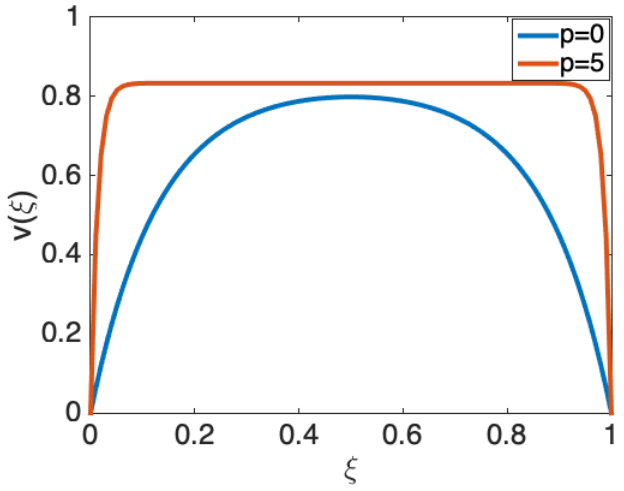


**Figure 6** Comparing the optimal velocity on two trajectories characterized by $p = 0$ and $p = 5$ respectively, when $\kappa = 0$.

- *Finite difference discretization.* The finite difference method is applied to solve monolithically both equations in (25), 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 $\mathcal{L}_c$ representing the total cost:

$$\mathcal{L}_c = \sum_{i=1}^{n_t} \mathcal{C}_i \tag{26}$$

- *Lagrange Net.* This strategy uses the algorithm illustrated in figure 8. 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:

$$\mathcal{L}_{LN} = \int_0^T \left(\frac{\partial \mathcal{C}}{\partial x} - \frac{d}{dt}\frac{\partial \mathcal{C}}{\partial \dot{x}}\right)^2 + \left(\frac{\partial \mathcal{C}}{\partial y} - \frac{d}{dt}\frac{\partial \mathcal{C}}{\partial \dot{y}}\right)^2 dt. \tag{27}$$

- *Physics Informed Neural Network, PINN.* The PINN is employed to solve Eq. (25). 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, $\mathcal{L}_{\text{PINN}}$, reads:

$$\mathcal{L}_{\text{PINN}} = \int_0^T \left((\ddot{x} - x + 2\dot{y})^2 + (\ddot{y} - y - 2\dot{x})^2\right) dt \tag{28}$$

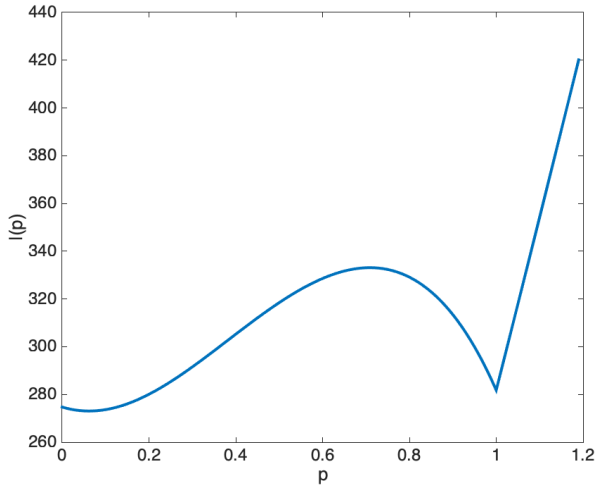


**Figure 7** Cost evolution $\mathcal{I}(p)$ associated with the cost function depicted in equation (23).

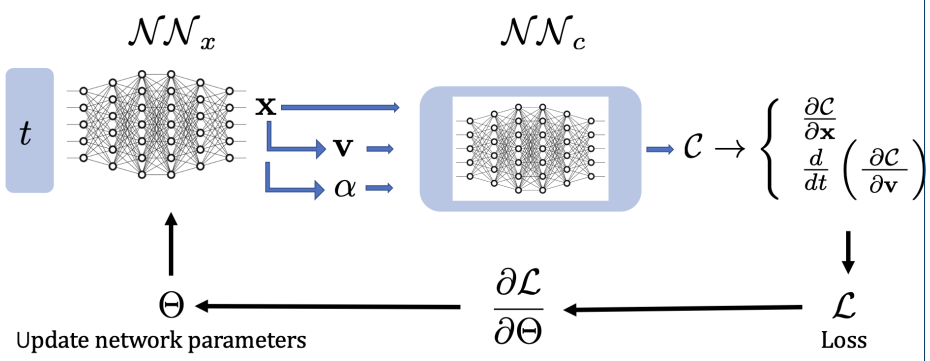


**Figure 8** Flow chart for the training of network $\mathcal{NN}_x$, when the cost function $\mathcal{C}$ is approximated by a neural network based regression $\mathcal{NN}_c$. The weights and biases of $\mathcal{NN}_x$ are noted $\Theta$.

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.

### 4.1 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. 9 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. 9 is computed as the sum of the cost $\mathcal{C}$ over all the time steps, as

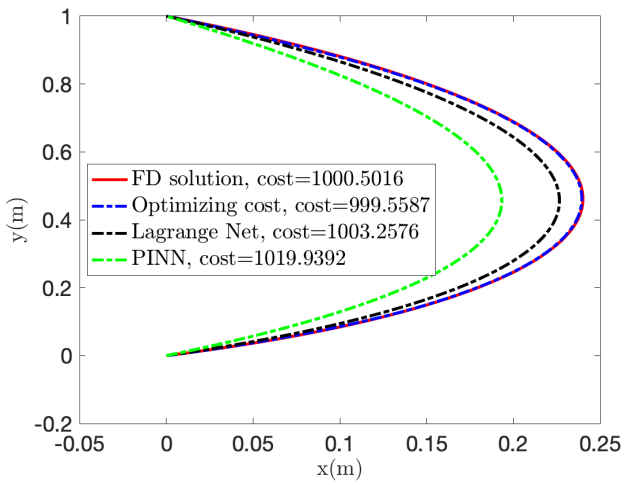


**Figure 9** Solution for $\mathcal{C} = (v_x + y)^2 + (v_y - x)^2$, $H = 1m$ and $T = 1s$ when using finite difference (FD), optimizing the cost directly, the Lagrange-Net flow chart illustrated in figure 8, and a Physics Informed Neural Network (PINN) solving the equations illustrated in 25.

| 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 $(= 1)$ | $linear$ |
| 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 $(= 1)$ | $linear$ |

**Table 1** The neural network used as the surrogate model $\mathcal{NN}_x$. Independent networks are used for predicting the coordinates $x$ and $y$, while having the same structure. $tanh$ stands for the hyperbolic tangent.

expressed in equation (26). The fact that the optimized cost is slightly smaller that the reference one can be explained by the fact that the former uses an approximation of the cost, that is, it proceeds from $\mathcal{NN}_c$.

### 4.2 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. 10.

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 Figure 11 proves. The associated velocities all along the trajectory, are compared for the horizontal and vertical velocity components in Figs. 12 and 13 respectively.

| 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 | $relu$ |
| 3 | Dense layer with 60 neurons | $relu$ |
| 4 | Dense layer with 60 neurons | $relu$ |
| 5 | Dense layer with 60 neurons | $relu$ |
| 6 | Output layer having the size of $x$ per node $(=1)$ | $linear$ |
| 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 | $relu$ |
| 3 | Dense layer with 60 neurons | $relu$ |
| 4 | Dense layer with 60 neurons | $relu$ |
| 5 | Dense layer with 60 neurons | $relu$ |
| 6 | Output layer having the size of $y$ per node $(=1)$ | $linear$ |

**Table 2** The deeper neural network used as the surrogate model $\mathcal{NN}_x$. Independent networks are used for predicting the coordinates $x$ and $y$, while having the same structure. $relu$ stands for the rectified linear unit.

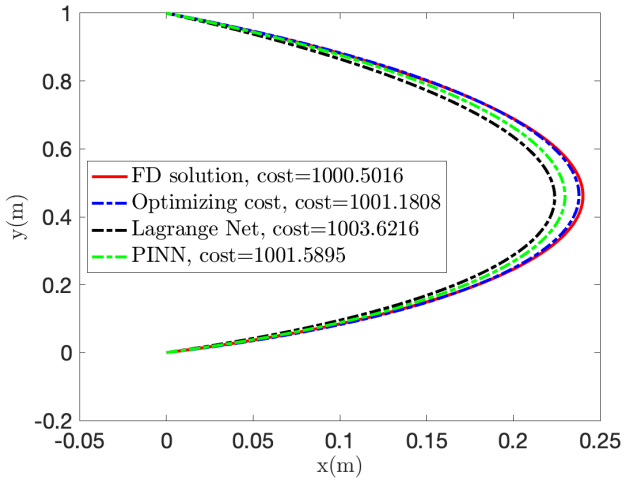


**Figure 10** Trajectory solutions when using the deeper network illustrated in table 2.

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.

## 5 Stochastic setting

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 $\mathcal{C}_d$, scaling with

$$\mathcal{C}_d \sim (v_d - v)^2 v_d, \tag{29}$$

| 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 60 neurons | $relu$ |
| 4 | Dense layer with 60 neurons | $relu$ |
| 5 | Dense layer with 60 neurons | $relu$ |
| 6 | Output layer having the size of $x$ per node $(=1)$ | $linear$ |
| 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 60 neurons | $relu$ |
| 4 | Dense layer with 60 neurons | $relu$ |
| 5 | Dense layer with 60 neurons | $relu$ |
| 6 | Output layer having the size of $y$ per node $(=1)$ | $linear$ |

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

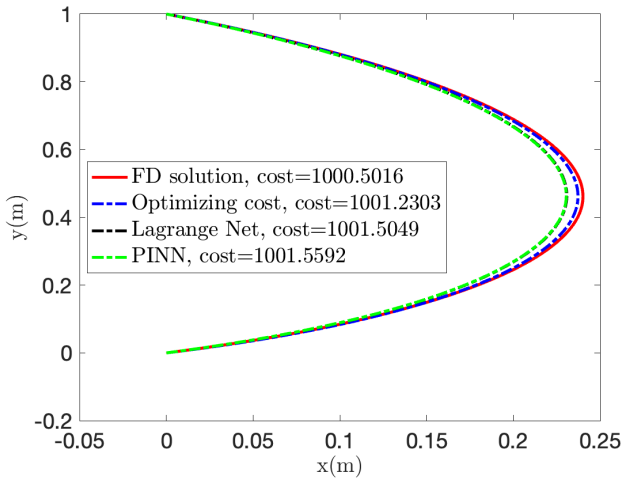


**Figure 11** Trajectory solutions when using the deeper network illustrated in table 3, combining $relu$ and $tanh$ activation functions.

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.

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 $v_d^{\texttt{nom}} = L/T$, 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

$$\mathcal{C}_t \sim (v_d T - L)^2. \tag{30}$$

Both contributions are combined into the global cost function

$$\mathcal{C} = \mathcal{C}_d + \mathcal{C}_t = (v_d - v)^2 v_d + \lambda (v_d T - L)^2, \tag{31}$$

with $\lambda$ a coefficient weighting both contributions.

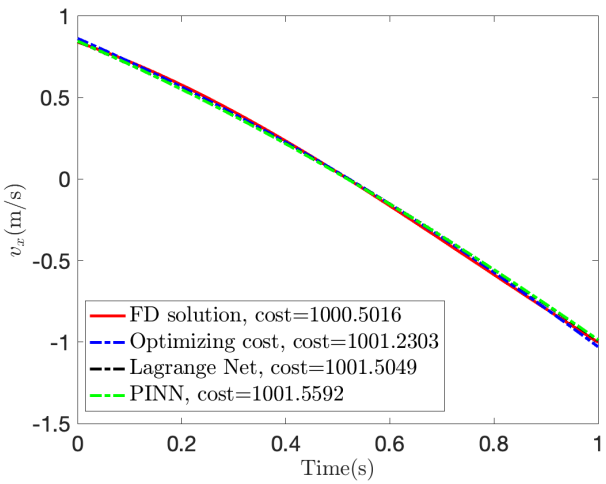


**Figure 12** Optimal velocities $v_x$ when using the deeper network illustrated in table 3, combining $relu$ and $tanh$ activation functions.

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:

$$\frac{\partial \mathcal{C}}{\partial v_d} = 0, \tag{32}$$

leading to

$$3v_d^2 - (4v - 2\lambda T^2)v_d + v^2 - 2\lambda TL = 0, \tag{33}$$

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. (33), whose associated cost $\mathcal{C}(v)$ results from applying Eq. (31).

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

$$\overline{\mathcal{C}} = \int_{v_{\texttt{min}}}^{v_{\texttt{max}}} \mathcal{C} \frac{1}{l} dv, \tag{34}$$

then, compute the wind velocity $v^*$ related to $\overline{\mathcal{C}}$, i.e. $\mathcal{C}(v^*) = \overline{\mathcal{C}}$, and finally the drone velocity associated to $v^*$ according to Eq. (31), that represents the optimal constant drone velocity.

This procedure can of course be generalized for more complex scenarios.

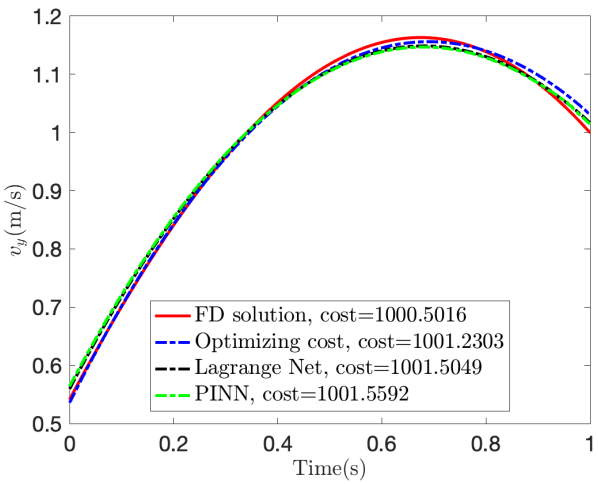


**Figure 13** Optimal velocities $v_y$ when using the deeper network illustrated in table 3, combining $relu$ and $tanh$ activation functions.

## 6 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.

The approach leads to a path optimization if the cost function $\mathcal{C}$ is either explicitly known or not.

We considered three different strategies for computing the optimal trajectory that were compared with the one obtained by using the finite difference method, considered as reference.

## Declarations

**Availability of data and material**
Data will be made available upon request.

**Competing interests**
The authors declare no competing interests.

**Author's contributions**
All the authors participated equally in the paper content.

**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 Sklodowska-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 details**
[1] PIMM Lab & SKF Chair, Arts et Metiers Institute of Technology, Paris, France. [2] PIMM Lab & ESI Chair, Arts et Metiers Institute of Technology, Paris, France. [3] LAMPA Lab & ESI Chair, Arts et Metiers Institute of Technology, Angers, France. [4] CNRS@CREATE CNRS, Singapore, Singapore. [5] ESI Group Chair, Aragon Institute of Engineering Research, Universidad de Zaragoza, Zaragoza, Spain.

**References**


1. 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). doi:10.23919/ACC.2019.8814905
2. Guanetti, J., Kim, Y., Borrelli, F.: Control of connected and automated vehicles: State of the art and future challenges. Annual Reviews in Control **45**, 18–40 (2018). doi:10.1016/j.arcontrol.2018.04.011
3. Lipp, T., Boyd, S.P.: Minimum-time speed optimisation over a fixed path. International Journal of Control **87**, 1297–1311 (2014)
4. 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). doi:10.1109/VPPC.2016.7791737
5. Sforza, A., Lenzo, B., Timpone, F.: A state-of-the-art review on torque distribution strategies aimed at enhancing energy efficiency for fully electric vehicles with independently actuated drivetrains. International Journal of Mechanical Control **20**, 3–15 (2019)
6. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research **30**(7), 846–894 (2011). doi:10.1177/0278364911406761. https://doi.org/10.1177/0278364911406761
7. Sachenbacher, M., Leucker, M., Artmeier, A., Haselmayr, J.: Efficient energy-optimal routing for electric vehicles. In: AAAI, pp. 1402–1407 (2011)
8. Sutton, R.S., Barto, A.G.: Reinforced learning. an introduction. The MIT Press
9. Kim, H., Pyeon, H., Park, J.S., Hwang, J.Y., Lim, S.: Autonomous vehicle fuel economy optimization with deep reinforcement learning. Electronics **9**(11) (2020). doi:10.3390/electronics9111911
10. 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 - S, 0–0 (2023). doi:10.3934/dcdss.2023080
11. Chinesta, F., Huerta, A., Rozza, G., Willcox, K.: Model order reduction. In the Encyclopedia of Computational Mechanics, Second Edition (2015)
12. Chinesta, F., Cueto, E., Abisset-Chavanne, E., Duval, J.L., Khaldi, F.E.: Virtual, digital and hybrid twins: A new paradigm in data-based engineering and engineered data. Archives of Computational Methods in Engineering **27**, 105–134 (2020)
13. Sancarlos, A., Champaney, V., Cueto, E., Chinesta, F.: Regularized regressions for parametric models based on separated representations. Adv. Model. and Simul. in Eng. Sci. **4**, 10 (2023)
14. Chinesta, F., Cueto, E.: Empowering engineering with data, machine learning and artificial intelligence: a short introductive review. Adv. Model. and Simul. in Eng. Sci. **9**, 21 (2023)
15. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics **378**, 686–707 (2019). doi:10.1016/j.jcp.2018.10.045
16. Quaranta, G., Haug, E., Duval, J.L., Chinesta, F.: International journal of material forming. Parametric evaluation of part distortion in additive manufacturing processes **13**, 29–41 (2020)