OPTIMAL VELOCITY PLANNING BASED ON THE SOLUTION OF THE EULER-LAGRANGE EQUATIONS WITH A NEURAL NETWORK BASED VELOCITY REGRESSION

Chady Ghnatios[1], Daniele di Lorenzo[2], Victor Champaney[2], Elías Cueto[3] and Francisco Chinesta[*,2,4]

[1]SKF Chair @ PIMM Laboratory, Arts et Métiers Institute of Technology
151 Boulevard de l'Hôpital, F-75013 Paris, France

[2]ESI GROUP Chair @ PIMM Laboratory, Arts et Métiers Institute of Technology
151 Boulevard de l'Hôpital, F-75013 Paris, France

[3]Aragon Institute of Engineering Research, Universidad de Zaragoza, Zaragoza, Spain

[4]CNRS@CREATE LTD, 1 Create Way, 08-01 CREATE Tower, 138602, Singapore

Abstract. Trajectory optimization is a complex process that includes an infinite number of possibilities and combinations. This work focuses on a particular aspect of the trajectory optimization, related to the optimization of a velocity along a predefined path, with the aim of minimizing power consumption. To tackle the problem, a functional formulation and minimization strategy is developed, by means of the Euler-Lagrange equation. The minimization is later performed using a neural network approach. The strategy is deemed Lagrange-Net, as it is based on the minimization of the energy functional, by the means of Lagrange's equation and neural network approximations.

1. Introduction.

With an eye towards autonomous vehicles, new ways to achieve fuel consumption optimization are in sight [8]. Several studies highlighted the need to achieve optimal power consumption control, for fuel cost reduction, but also as a means of reduction of CO_2 emissions also known as eco-driving [1, 6]. The problem of fuel consumption reduction by means of velocity optimization consists of a particular case of path optimization problems, where the path is previously defined, and the objective function aims to optimize the velocity profile along the path [9].

The problem of optimal velocity planning is tackled by several authors, while assuming a known path, to achieve minimum travel time subjected to environmental and configuration constraints [9]. For example, in [3] and [12], the authors focus on the torque distribution over the tire and the moment generation effect to optimize energy consumption. Motion optimization using sampling algorithms such as random sampling and random forests are described in [7], while the authors in [11] use an A^* search algorithm to optimize the consumption and battery life in autonomous electric vehicles. More recent approaches focus on the use of reinforcement learning approach for optimal velocity estimation and fuel consumption reduction [8].

2020 *Mathematics Subject Classification.* Primary: 37N35; Secondary: 35G61.

Key words and phrases. Machine learning, artificial intelligence, optimal planning, Euler-Lagrange equations.

[*]Corresponding author: Francisco Chinesta.

The present work explores a procedure based on the use of physics-based algorithms coupled to neural networks, to solve the energy consumption optimization problem, inspired of the Physics Informed Neural Networks –PINN– [10] while using the associated Euler-Lagrange equations.

In Section 2 we first review the functional formulation of the energy consumption, along with the derivation of the local differential equation minimizing it. Section 3 presents and discusses some numerical examples on a given consumption cost function. Finally, Section 4 addresses some concluding remarks.

2. **Consumption minimization and functional formulation.** We assume that the fuel consumption (or, equivalently, charge consumption for electrical cars) is expressed as a cost function $\mathcal{C}$ that depends on the position along the trajectory, represented by the curvilinear coordinate s, $s \in \Gamma$, $\Gamma = [0, L]$, with L the trajectory length, on the velocity at each position on the trajectory $v(s)$, with $v(s = 0) = v(s = L) = 0$, and on its rate of change v' with respect to the position s, $v'(s) = \frac{dv(s)}{ds}$.

It is important to note that $v'(s)$ does not represent the acceleration. The velocity change has an impact in the consumption, but the later is not a direct function of the acceleration. For this reason v' is integrated in the cost function as a penalty to enforce a constant velocity.

The slope in the case of a road, or any other environmental resistance (wind, ...) can be defined as a function of that coordinate s. Thus, once a consumption cost function $\mathcal{C}(s, v, \frac{dv}{ds}) \equiv \mathcal{C}(s, v(s), v'(s))$ is available, one may wish to minimize the total consumption (in an analogous way the action functional is defined in Lagrangian mechanics) along the path using:

$$\mathcal{I}(v) = \int_0^L \mathcal{C}\left(s, v, v'\right)\, ds, \tag{1}$$

with $\mathcal{I}(v)$ the functional to minimize.

The optimization problem becomes now a minimization of the functional $\mathcal{I}(v)$, whose minimum can be found from

$$\delta \mathcal{I}(v) = 0. \tag{2}$$

By using the associated Euler-Lagrange equation [4, 5]

$$\frac{\partial \mathcal{C}}{\partial v} - \frac{d}{ds}\frac{\partial \mathcal{C}}{\partial v'} = 0, \tag{3}$$

the optimal velocity is the one that results from the solution of Eq. (3).

The problem defined in Eq. (3) can be solved numerically when the analytical expression of $\mathcal{C}(s, v, v')$ is available, by using finite elements or finite differences, for instance. However, sometimes the analytical expression of $\mathcal{C}(s, v, v')$ is not available. This may require a surrogate model using neural networks, for example. In such a case, we propose using a neural network to solve the optimization problem and find the optimal velocity $v(s)$ along the trajectory.

3. **Numerical examples.** In this section, for illustration purposes, and without any loss of generality, we consider the local cost function $\mathcal{C}(s, v, v')$ having the following form:

$$\mathcal{C} = av^2 + b\alpha(s)v^2 + c\left(v'(s)\right)^2 + \lambda\left(v - v_{\max}\right)^2, \tag{4}$$

where a, b, c and λ are four parameters, the first expressing the drag, the second the slope or other environmental effects modeled by $\alpha(s)$ (term that could be positive or negative, resisting or favoring the motion), and the last two penalty terms, the former enforcing a constant velocity and the later enforcing a velocity ensuring the fastest travel.

In what follow we start by introducing the optimization process operating on the analytical cost function defined in Eq. (4). Then, in a second stage, we will fit the cost function using a regression and attempt to minimize the total cost without any previous knowledge of the analytical form of the associated local cost function. The fact of having the analytical form of $\mathcal{C}$ given by Eq. (4), enables the integration of the Euler-Lagrange equation by using the finite difference method, to obtain the so-called "ground truth" numerical solution, to which we can compare the output of the neural network to quantify the results accuracy.

3.1. **Lagrange-Net optimization using the analytical formulation of the local consumption $\mathcal{C}$.** In this section we consider the velocity control neural network $\mathcal{H}$ has access to the analytical form of the local cost function given by Eq. (4). The neural network will attempt to solve Eq. (3) in $s \in \Gamma = [0, L]$, with the following boundary conditions:

$$\begin{cases} v(s=0) = 0, \\ v(s=L) = 0. \end{cases} \tag{5}$$

Boundary conditions are imposed using the following change of variables

$$v(s) = \gamma(s) \cdot y(s) + \psi(s), \tag{6}$$

with $\gamma(s)$ vanishing on the boundary conditions and $\psi(s)$ satisfying the required boundary conditions. In the present case we consider $\psi(s) = 0$ and $\gamma(s) = s(L - s)/200$.

The output of the neural network $\mathcal{H}$ is therefore $y(s)$ instead of $v(s)$. A flow chart is illustrated in Fig. 1. The partial derivatives of the cost function are obtained by automatic differentiation of the neural network surrogate model $\mathcal{H}$ using the tensorflow function `GradientTape`. It is important to note that partial derivatives are computed straightforward, however, the total derivative acting on $\partial\mathcal{C}/\partial v'$, needs to be transformed by using the chain rule. By introducing the notation $\hat{\mathcal{C}}(s, v, v') \equiv \partial\mathcal{C}/\partial v'$, the total derivative reads

$$\frac{d\hat{\mathcal{C}}}{ds} = \frac{\partial\hat{\mathcal{C}}}{\partial s} + \frac{\partial\hat{\mathcal{C}}}{\partial v}\frac{dv}{ds} + \frac{\partial\hat{\mathcal{C}}}{\partial v'}\frac{dv'}{ds}. \tag{7}$$

A custom neural network training loop is created for this problem, the adopted parameters are $L = 30\ km$, $v_{\max} = 1\ m/s$, $N = 200$ sampling point in the domain Γ, and

$$\alpha = \cos(\pi s/L), \tag{8}$$

representing a decreasing positive slope (resistance) in $[0, L/2]$ and then, an increasing negative slope in $[L/2, L]$, as depicted in Fig. 2.

The adopted network architecture is described in Table 1. The loss $\mathcal{L}$ function to be minimized is the sum of square errors of the Euler-Lagrange equation residual

$$\mathcal{L} = \sum_{i=1}^{i=N} \left(\left.\frac{\partial C}{\partial v}\right|_{s_i} - \frac{d}{ds}\left.\frac{\partial C}{\partial v'}\right|_{s_i} \right)^2. \tag{9}$$

This formulation bears some similitudes with the Lagrangian neural networks by Cranmer et al. [2], although in our case the Lagrangian is replaced by the cost function, whose precise form could be known in advance or determined from data.

Layers	Shape	activation
1	Input data layer, shape $(N \times 1)$	No activation
2	Reshape layer into a 3D Tensor of shape $(N \times 1 \times 1)$	No activation
3	2D convolution, 30 filters, kernel (5×1), strides (5×1), no padding	elu
4	Flatten layer	No activation
5	Dense fully connected layer, 1200 units	elu
6	Reshape layer into a 3D Tensor of shape $(120 \times 10 \times 1)$	No activation
7	2D convolution transpose, 60 filters, kernel (5×1), strides (5×1), no padding	elu
8	2D convolution transpose, 30 filters, kernel (5×1), strides (5×1), no padding	elu
9	2D convolution transpose, 5 filters, kernel (4×1), strides (4×1), no padding	linear
10	Flatten layer	No activation
11	Dense fully connected layer, N units	linear

TABLE 1. Structure of the deep convolution/convolution transpose neural network used for the fitting of $\mathcal{H}$. "elu" stands for the exponential linear unit, while "linear" stands for linear activation or no activation function.

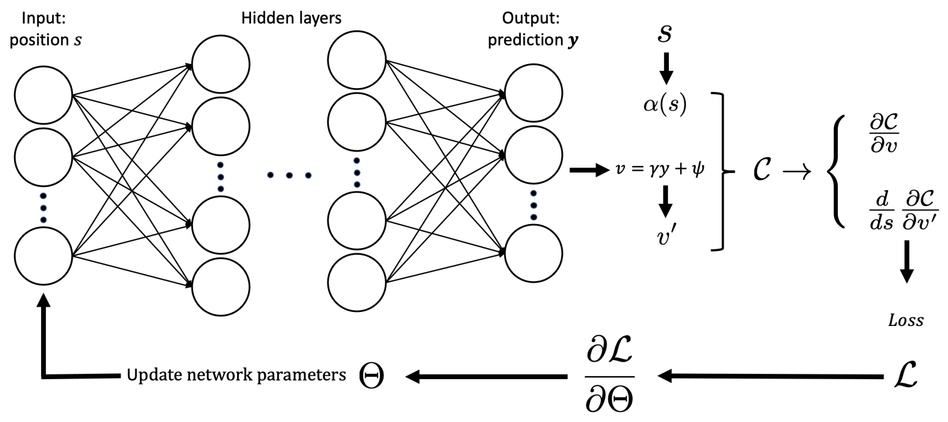

Input:
position s
Hidden layers
Output:
prediction y
s
$\alpha(s)$
$v = \gamma y + \psi$
v'
$\mathcal{C} \rightarrow$
$\frac{\partial \mathcal{C}}{\partial v}$
$\frac{d}{ds}\frac{\partial \mathcal{C}}{\partial v'}$
Loss
$\mathcal{L}$
$\frac{\partial \mathcal{L}}{\partial \Theta}$
Update network parameters Θ

FIGURE 1. Flow chart for the training of the control network $\mathcal{H}$, when having access to the analytical form of the cost function $\mathcal{C}$. The network's weights and biases are named Θ, while $v' = \frac{dv}{ds}$.

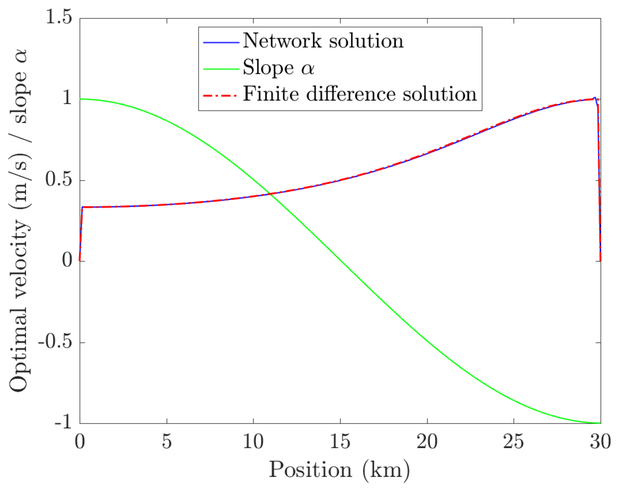


FIGURE 2. Neural network solution of the optimal velocity for $a = 1$, $b = 1$, $c = 0$ and $\lambda = 1$, along with the corresponding finite difference solution for the same number of nodes N in the domain Γ.

Results are presented in Fig. 2 for $a = 1$, $b = 1$, $c = 0$, and $\lambda = 1$ and in Fig. 3 for $a = 10$, $b = 10$, $c = 1$ and $\lambda = 10$, respectively. The prediction accuracy is very good in bot cases, with a mean absolute percentage error (MAPE) of 0.35% in the former and of 1.19% in the last.

3.2. **Lagrange-Net optimization using the regression of the local consumption $\mathcal{C}$.** In this section, we start by creating a surrogate model of the local cost function. For instance, we are building a regression surrogate model $\mathcal{G}$ that

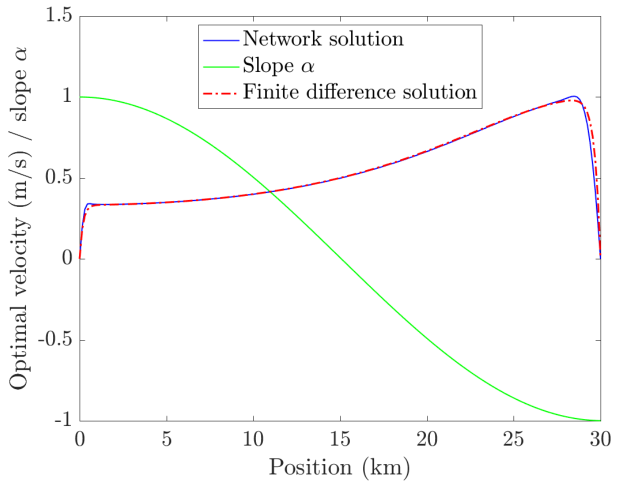
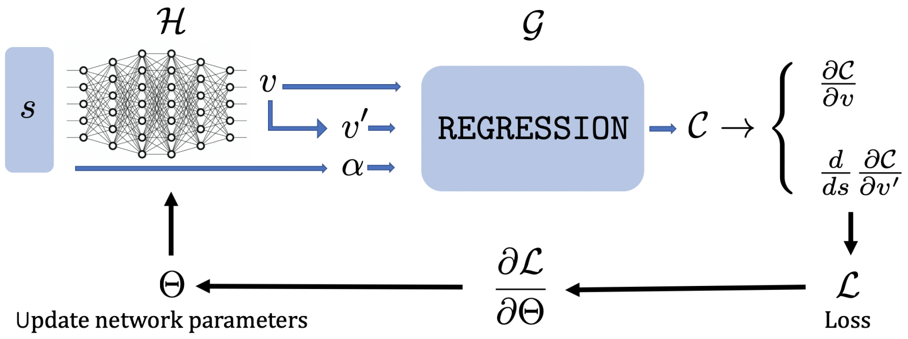


FIGURE 3. Neural network solution of the optimal velocity for $a = 10$, $b = 10$, $c = 1$ and $\lambda = 10$, along with the corresponding finite difference solution for the same number of nodes N in the domain Γ.



$\mathcal{H}$
$\mathcal{G}$
s
v
v'
α
REGRESSION
$\mathcal{C} \rightarrow$
$\frac{\partial \mathcal{C}}{\partial v}$
$\frac{d}{ds}\frac{\partial \mathcal{C}}{\partial v'}$
$\mathcal{L}$
Loss
$\frac{\partial \mathcal{L}}{\partial \Theta}$
Θ
Update network parameters

FIGURE 4. Flow chart for the training of the control network $\mathcal{H}$, when the cost function $\mathcal{C}$ is approximated by a regression model $\mathcal{G}$. The weights and biases of $\mathcal{H}$ are named Θ, while $v' = \frac{dv}{ds}$.

can reproduce the local cost function $\mathcal{C}$ given in Eq. (4). The general flow chart of the algorithm is now illustrated in Fig. 4.

In absence of experimental data, the surrogate model of the local cost, $\mathcal{G}$, is obtained by sampling the known local cost function given in Eq. (4).

In what follows, we use a polynomial regression to fit $\mathcal{G}$. To this end, we start by creating a database of 10.000 velocity fields, all built by using polynomials of degrees 1 to 10 (1000 polynomials for each degree) with the coefficients randomly chosen in $[-10, 10]$. Then, 200 points are distributed in Γ, where the 10.000 polynomials are particularized and then the local cost function evaluated from Eq. (4), for a given

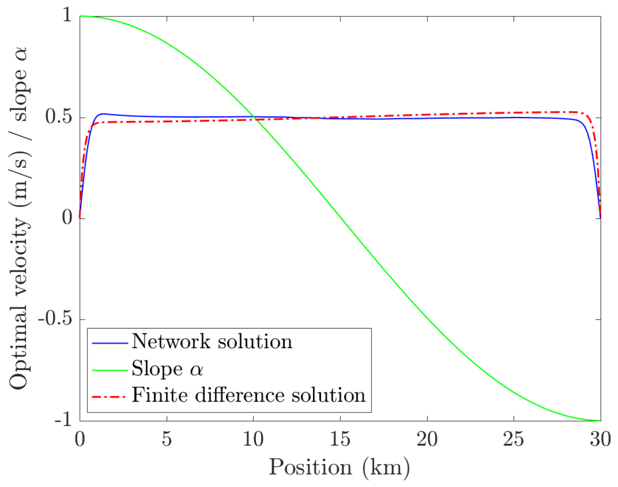


FIGURE 5. Neural network solution of the optimal velocity for $a = 10$, $b = 1$, $c = 1$ and $\lambda = 10$, along with the corresponding finite difference solution for the same number of nodes N in the s domain. The cost is approximated using the regression $\mathcal{G}$.

choice of parameters a, b, c and λ. A regression is performed for each value of α, that is, 200 polynomial regression in the present case.

Each regression proceeds from the matrix formulation

$$X = \begin{bmatrix} | & | & | & | & | \\ v & v^2 & v' & (v')^2 & 1 \\ | & | & | & | & | \end{bmatrix}. \tag{10}$$

with 80% of the data is selected for training the algorithm, 20% for testing.

The fitting weights Υ of $\mathcal{G}$ are obtained using:

$$\Upsilon = \left(X^T X\right)^{-1} X^T Y, \tag{11}$$

with Y being the cost vector.

The cost function mean relative errors (mean absolute errors –MAE) are about 10^{-10} for both the training and the testing sets.

Once $\mathcal{G}$ is available, it is set as non trainable and the training of $\mathcal{H}$ can start. The same loss function and neural network shape illustrated in Section 3.1 are considered here. Moreover, the same boundary conditions are used, and imposed using the same change of variables. The results are illustrated in Fig. 5 for $a = 10$, $b = 1$, $c = 1$ and $\lambda = 10$. Figure 6 for $a = 10$, $b = 10$, $c = 1$ and $\lambda = 10$. Again, the accuracy in the performed predictions seems good enough, with a MAPE of 5.72% in the former case, and of 1.09% in the last case.

3.3. **Lagrange-Net optimization using a neural network approximation of the local consumption $\mathcal{C}$.** Similarly to section 3.2, we start by creating a surrogate model of the local cost function. However, in the present case the regression $\mathcal{G}$, here referred as $\mathcal{G}_2$, representing the cost function $\mathcal{C}$ is obtained by using a neural network, that is trained offline. The general flow chart is now illustrated in Fig. 7.

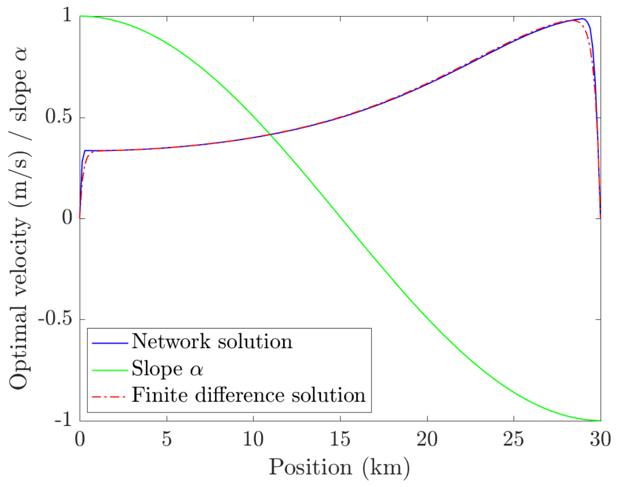


FIGURE 6. Neural network solution of the optimal velocity for $a = 10$, $b = 10$, $c = 1$ and $\lambda = 10$, along with the corresponding finite difference solution for the same number of nodes N in the s domain. The cost is approximated using the regression $\mathcal{G}$

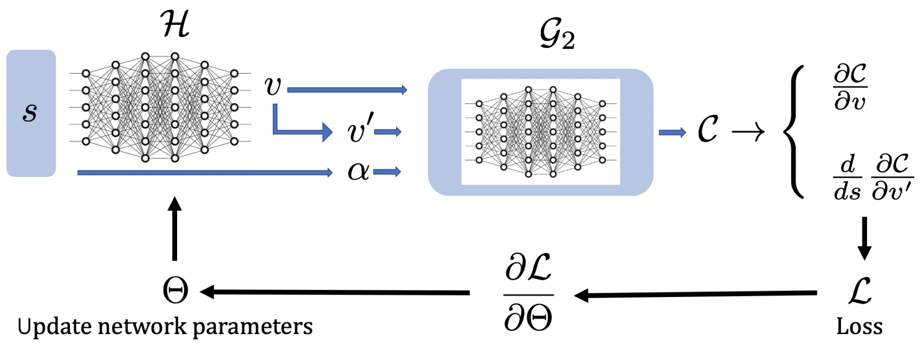

$\mathcal{H}$
s
v
v'
α
$\mathcal{G}_2$
$\mathcal{C} \rightarrow$
$\frac{\partial \mathcal{C}}{\partial v}$
$\frac{d}{ds}\frac{\partial \mathcal{C}}{\partial v'}$
$\mathcal{L}$
Loss
$\frac{\partial \mathcal{L}}{\partial \Theta}$
Θ
Update network parameters

FIGURE 7. Flow chart for the training of the control network $\mathcal{H}$, when the cost function $\mathcal{C}$ is approximated by a neural network based regression $\mathcal{G}_2$. The weights and biases of $\mathcal{H}$ are named Θ, while $v' = \frac{dv}{ds}$.

To efficiently construct and train the neural network used as a surrogate model, $\mathcal{G}_2$, the inputs are naturally the velocity v and its derivative v'. Moreover, just after the input layers, a custom layer is build using `Keras` library, which computes v^2 and $(v')^2$ point-wise. Once all the relevant inputs are available, a concatenation is followed by a single fully connected layer to compute the regression output.

The model $\mathcal{G}_2$ is trained using `ADAM` stochastic gradient descent algorithm, with an adaptive learning rate, defined using a custom loop. The converged $\mathcal{G}_2$ can approximate the cost function $\mathcal{C}$ with a MAPE of $1.26 \times 10^{-4}\%$ on the training set and $1.27 \times 10^{-4}\%$ on the testing set. The same data described in section 3.2 is used to train $\mathcal{G}_2$.

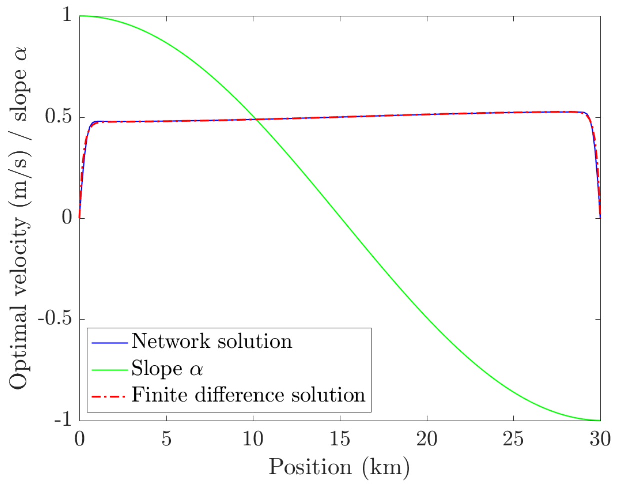


FIGURE 8. Neural network solution of the optimal velocity for $a = 10$, $b = 1$, $c = 1$ and $\lambda = 10$, along with the corresponding finite difference solution for the same number of nodes N in the s domain. The cost is approximated using the neural network surrogate model $\mathcal{G}_2$.

The parameters of the surrogate model $\mathcal{G}_2$ are now set as *not trainable*, and $\mathcal{G}_2$ is embedded in the training loop of $\mathcal{H}$, as illustrated in figure 7. The same loss function, neural network shape and boundary conditions imposition is used to train $\mathcal{H}$, as described in Section 3.1. The optimal velocity output of $\mathcal{H}$ is finally passed into a shallow Laplacien smoothening filter to flatten any tiny noise still existing in the velocity solution. The results illustrated in Fig. 8 represent the solution for $a = 10$, $b = 1$, $c = 1$ and $\lambda = 10$, while the ones in Fig. 9 correspond to $a = 10$, $b = 10$, $c = 1$ and $\lambda = 10$. The accuracy in the performed predictions is good, with a MAPE of 1.433% in the former case, and of 1.445% in the last case.

4. **Conclusion.**

In this work proves the feasibility of training a neural network control to prescribe the optimal velocity profile in a predefined path. The network involves two parts, the one that defines the local cost function and the one that optimizes the velocity.

Such cost function can be defined analytically or obtained by using regressions operating on experimental data. The cost is differentiated to obtain the Euler-Lagrange formulation of the problem, that serves to compute the loss function.

The proposed neural network is able to reproduce the solution with high fidelity, if compared to the finite-difference, ground truth solution. The obtained results are also a demonstration that such an approach can be used to optimize the velocity profile in any moving object, such as cars or drones, along a predefined path.

More complex costs or optimality criteria will require specific treatments. Moreover, more general scenarios could involved parametrized trajectories, that lead to parametrized Euler-Lagrange equations which can be efficiently treated within the PGD framework or the neural setting just proposed. The most general formulation consisting on looking for the optimal trajectory $\mathbf{x}(t)$ and a cost function involving

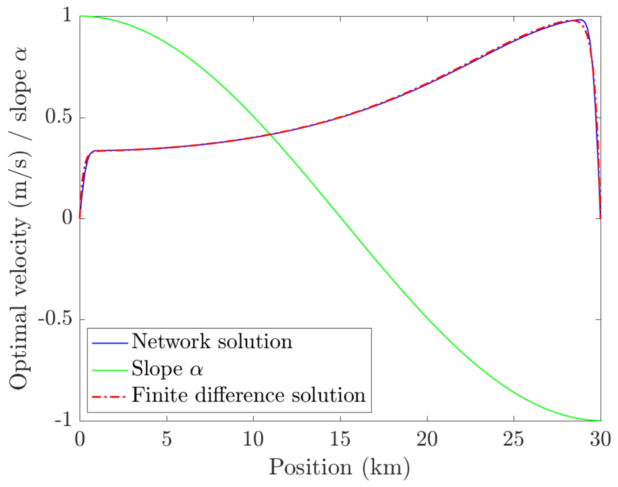


FIGURE 9. Neural network solution of the optimal velocity for $a = 10$, $b = 10$, $c = 1$ and $\lambda = 10$, along with the corresponding finite difference solution for the same number of nodes N in the s domain. The cost is approximated using the neural network surrogate model $\mathcal{G}_2$

the tile, the velocity and the acceleration, constitute a work in progress, with some first promising results.

Acknowledgments. This research is part of the programme DesCartes and is supported by the National Research Foundation, Prime Minister Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

This work was supported by the European Union Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 956401 (XS-Meta) and 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 project UZ-2019-0060, the ESI Group and SKF chairs at Arts et Metiers Institute of Technology, as well as Renault Group through the *Plan Relance*.

REFERENCES

[1] S. Bae, Y. Kim, J. Guanetti, F. Borrelli and S. Moura, Design and implementation of ecological adaptive cruise control for autonomous driving with communication to traffic lights, in: *2019 American Control Conference (ACC)*, (2019), 4628-4634.
[2] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel and S. Ho, Lagrangian neural networks, arXiv preprint, (2020). `arXiv:2003.04630`.
[3] G. De Filippis, B. Lenzo, A. Sorniotti, K. Sannen, J. De Smet and P. Gruber, On the energy efficiency of electric vehicles with multiple motors, in: *2016 IEEE Vehicle Power and Propulsion Conference (VPPC)*, (2016), 1-6.
[4] C. Fox, *An Introduction to the Calculus of Variations*, Courier Dover Publications, 1987.
[5] H. Goldstein, C. Poole and J. Safko, *Classical Mechanics*, 3rd Edition, Addison Wesley., 2014.

[6] J. Guanetti, Y. Kim and F. Borrelli, Control of connected and automated vehicles: State of the art and future challenges, *Annual Reviews in Control*, **45** (2018), 18-40.

[7] S. Karaman and E. Frazzoli, Sampling-based algorithms for optimal motion planning, *The International Journal of Robotics Research*, **30** (2011), 846-894.

[8] H. Kim, H. Pyeon, J. S. Park, J. Y. Hwang and S. Lim, Autonomous vehicle fuel economy optimization with deep reinforcement learning, *Electronics*, **9** (2020).

[9] T. Lipp and S. Boyd, Minimum-time speed optimisation over a fixed path, *International Journal of Control*, **87** (2014), 1297-1311.

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

[11] M. Sachenbacher, M. Leucker, A. Artmeier and J. Haselmayr, Efficient energy-optimal routing for electric vehicles, in: *AAAI*, **25** (2011), 1402-1407.

[12] A. Sforza, B. Lenzo and F. Timpone, 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** (2019), 3-15.