POLITECNICO DI TORINO
Repository ISTITUZIONALE

Self-adaptive neural network model predictive anti-jerk control of electric powertrains

Original
Self-adaptive neural network model predictive anti-jerk control of electric powertrains / Frison, G., Alberti, F., Ciravegna, L., Dimauro, L., Sorniotti, A.. - In: MECHANISM AND MACHINE THEORY. - ISSN 0094-114X. - ELETTRONICO. - 214:(2025), pp. 1-25. [10.1016/j.mechmachtheory.2025.106082]

Availability:
This version is available at: 11583/3000727 since: 2025-06-06T08:03:55Z

Publisher:
Elsevier

Published
DOI:10.1016/j.mechmachtheory.2025.106082

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

19 November 2026

Mechanism and Machine Theory 214 (2025) 106082



Contents lists available at ScienceDirect

Mechanism and Machine Theory

journal homepage: www.elsevier.com/locate/mechmt



Research paper

Self-adaptive neural network model predictive anti-jerk control of electric powertrains

Gianluca Frison, Fabio Alberti, Luca Ciravegna, Luca Dimauro, Aldo Sorniotti *

Department of Mechanical and Aerospace Engineering (DIMEAS), Politecnico di Torino, 10129 Turin, Italy

ARTICLE INFO

Keywords:
Drivetrain dynamics
Anti-jerk control
Neural network
Model predictive control
Controller self-adaptation

ABSTRACT

The study introduces a proof-of-concept self-adaptive neural network model predictive control (SA-NNMPC) system, which uses a neural network as main component of the prediction model, for the anti-jerk control of electric vehicles. Through the adaptation mechanism of the network and cost function weights during vehicle operation, which is activated when the plant behaves significantly differently from its digital twin, the SA-NNMPC architecture adjusts to the progressive vehicle aging, or to the replacement of hardware parts, which is expected to be an important feature of next-generation vehicles. Validation tests and simulations show that the neural network accurately replicates the drivetrain dynamics of the considered electric vehicle, and, for nominal conditions, already leads to a performance improvement of the NNMPC implementation – which can run in real-time on a rapid control prototyping unit – with respect to a benchmarking nonlinear model predictive anti-jerk controller. Moreover, the preliminary simulation results confirm the potential of the proposed architecture in terms of: i) adaptability to operating conditions not covered in the original training, and variations of vehicle parameters; and ii) auto-tuning of the algorithm when applied to different vehicles.

1. Introduction

The last 15 years have seen major academic and industrial research efforts on innovative technologies to continuously improve the performance of electric vehicles (EVs), which experience increasing market penetration. The electrification trend has brought a variety of architectural solutions for electric powertrains, which can have an on-board or in-wheel layout, depending on whether the electric machine/s (EM/s) is/are located in the sprung or unsprung mass. In the former case [1], each electric powertrain is connected to the wheels through a mechanical transmission system, half-shafts, and constant-velocity joints.

In the on-board configurations, the presence of half-shafts implies non-negligible torsional dynamics during the powertrain torque transients, which result in corresponding wheel torque and longitudinal acceleration oscillations, with negative impact on component durability and the passenger comfort domain commonly referred to as drivability. The most influential vehicle hardware parameters on drivability performance are the mass moments of inertia of the rotating parts of the powertrain, as well as the torsional compliances [2] and mechanical plays (i.e., the backlash) [3,4] of the driveline components. Tip-in and tip-out manoeuvres, involving fast positive and negative variations of the torque demand, are typically used to assess the relevant dynamics during simulations or experiments. Although these aspects have been traditionally evaluated at the vehicle level through the subjective feedback of professional test drivers, the recent trend is to use objective key performance indicators (KPIs), e.g., the root mean square (RMS) value and vibration

* Corresponding author.
E-mail address: aldo.sorniotti@polito.it (A. Sorniotti).

https://doi.org/10.1016/j.mechmachtheory.2025.106082
Received 2 January 2025; Received in revised form 10 May 2025; Accepted 18 May 2025
Available online 3 June 2025
0094-114X/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

dose value (VDV) [5] of the frequency-weighted longitudinal acceleration a_x, or other indicators based on the time profiles of a_x and further relevant variables during tip-ins and tip-outs [6]. For example, the KPIs of the latter category include the rise time and overshoot magnitude of a_x, and the maximum magnitude of the jerk, i.e., the time derivative of a_x [7].

Despite being well-known in internal combustion engine (ICE) driven vehicles, the torsional drivetrain dynamics are significant also in EVs, which are characterised by the absence of dedicated mechanical components, such as clutch dampers and flywheels. To achieve desirable drivability, both ICE driven vehicles and EVs are equipped with anti-jerk controllers, which modify the human or automated driver torque demand, and attenuate the oscillations. Anti-jerk controllers address the first torsional mode [8] of the drivetrain, corresponding to a natural frequency in the $1 \div 15$ Hz range [9,10]. Electric powertrains are generally easier to control than their ICE counterparts, because of the more accurate torque generation capability [11,12] and the higher EM bandwidth with respect to (w.r.t.) ICEs [13]. The recent literature review on anti-jerk control in [14], with coverage until 2020, categorises the control formulations according to their feedforward or feedback nature, and in the latter case based on the considered anti-jerk error variable, which must be representative of the level of drivetrain oscillations or vehicle discomfort.

The most recent literature has broadened the anti-jerk control functionality. For example, reference [15] presents nonlinear model predictive controllers (NMPCs) that integrate the traction control and anti-jerk control functions, for EVs with centralised powertrains, benefitting from the preview of the tyre-road friction level, e.g., deriving from vehicle-to-everything (V2X) connectivity. Kock et al. [16] propose a controller that considers and attenuates the driveline oscillations during brake blending operation. The NMPCs in [17,18] use the information on the road profile ahead to compensate for the longitudinal acceleration oscillations induced by road irregularities, through the modulation of the torque demand of in-wheel and on-board electric powertrains. Ahmad et al. call anti-jerk controllers algorithms that attenuate the sprung mass motions caused by road irregularities, through the actuation of a semi-active suspension system [19], or aerodynamic surfaces installed on the vehicle body [20,21].

In the classical anti-jerk control definition, targeting the compensation of the torsional driveline dynamics, the main challenges are: i) to achieve desirable and consistent performance with the typical sampling times and time-varying delays of the on-board vehicle communication network [22]; and ii) to provide robustness during the vehicle lifecycle, w.r.t. the variation of the most relevant parameters, such as the drivetrain backlash and torque response delays of the powertrain [23]. Indeed, Figel's review [24] states that backlash is almost always considered in some form, during the design of drivability controllers based on the rotational drivetrain dynamics. In most applications, the effective half-shaft stiffness and backlash are identified at the beginning of the vehicle lifetime, see [25,26], and then are used for torque estimation and driveline oscillation control. Nevertheless, the literature includes examples of real-time driveline backlash estimation algorithms, e.g., see [3] and [27,28]. The algorithm in [29] for real-time backlash estimation and update is integrated into a clunk noise controller using a soft-landing reference governor [30]. In [31], a backlash estimator is integrated into an oscillation control strategy for a hybrid driveline, which is evaluated through simulations.

Thanks to the progressive enhancement of the computational capability of the available automotive control hardware as well as the introduction of computationally efficient solvers [32], the last years have experienced the growing application of model predictive control (MPC) algorithms to vehicle control functions, including anti-jerk control [33]. Moreover, recent automotive research – not applied to the driveline dynamics control problem yet – has proposed the integration of machine learning and artificial intelligence (AI) functionalities in MPCs, as discussed by Norouzi et al. [34]. In this field, one of the options is to partially or completely replace the physics-based prediction model of the MPC with a neural network (NN), to reduce computational cost [35] and improve prediction accuracy through multi-layer NN arrangements. The result is represented by neural network MPCs (NNMPCs), which have been applied to the path tracking control problem of automated vehicles [36–39], and the control of diesel engine emissions [40,41]. The recent literature has proposed physics-informed neural networks for the effective emulation of system dynamics [42,43], which in some cases have been used as prediction models in NNMPC implementations applied to various non-automotive systems, e.g., see the case studies in [44,45].

Another trend is represented by the continuous update of the control algorithms to adapt to the current condition or operating environment of the vehicle, which is a timely topic in the context of software-defined vehicles [46], collateral to this research. The literature includes examples of learning MPCs [47] for autonomous racing, which are used to progressively reduce the lap time of a vehicle on a track, based on the data from previous laps. In [48], an adaptive MPC strategy is proposed for a hybrid energy storage system, based on the real-time variation of the cost function weights via fuzzy logic rules. A fuzzy adaptation scheme is employed in [49] for the energy-efficient torque-vectoring control of an EV with multiple powertrains. Abtahi et al. [50] use an NN algorithm to vary the weights of an MPC strategy for energy-efficient adaptive cruise control, benefitting from the road grade preview. Porsche [51] has recently proposed a reinforcement learning approach for accelerated anti-jerk control calibration.

In summary, the analysis of the literature highlights that: i) only a limited number of research papers address the automotive application of NNMPC, without investigating the online adaptation of the NN prediction model, and related controller cost function weights; ii) there is no available NNMPC-based anti-jerk algorithm; and iii) when MPC is employed for anti-jerk control [33], the controller calibration is conducted only offline.

To address the identified gap, this study provides the following novel contributions to the subject area of the control of the torsional drivetrain dynamics:

- An NNMPC implementation of an anti-jerk controller for an EV with on-board powertrains, which outperforms – already for nominal conditions of the vehicle, simulated with an experimentally validated model – two considered benchmarking anti-jerk controllers, respectively using a physics-based nonlinear model predictive control approach and a tachometric approach.
- The preliminary proof-of-concept analysis of the potential drivetrain dynamics benefit of the online update, i.e., performed during vehicle operation, of the NN prediction model and cost function weights, to address: i) different operating conditions w.r.t those of

the initial offline training; ii) variations of vehicle parameters, e.g., transmission backlash, powertrain torque time constant, and tyre characteristics; and iii) controller auto-tuning, i.e., the direct application of a controller originally calibrated for a specific EV to a different one.
- The demonstration of the real-time implementability of the NNMPC anti-jerk approach, with the trade-off analysis between computational effort and performance, which highlights its potential practical feasibility for real-vehicle implementation.

The remainder is organised as follows: Section 2 describes the simulation and control environment, and the case study vehicle; Section 3 presents the proposed NNMPC and NMPC formulations; Section 4 discusses the online self-adaptive control architecture; Section 5 presents the simulation results, obtained on an experimentally validated vehicle model; finally, Section 6 summarises the main conclusions.

2. Experimentally validated simulation environment and case study electric vehicle

2.1. Simulation and control architecture

The simulation and control environment, reported in Fig. 1, includes:

- The human or automated driver model, defining the reference powertrain torque, T_{ref}.
- The NNMPC anti-jerk algorithms, which represent a novelty of the study, embedding an internal (or prediction) model characterised by the combination of NN-based and physics-based formulations, and generating the corrective anti-jerk control torque, T_{corr}, through the online solution of an optimal control problem (OCP), starting from T_{ref} and the relevant measured or estimated variables. The proposed distributed architecture has an independent anti-jerk controller for each electric powertrain. Here and in the remainder, for simplicity, the notations omit the reference to any specific EV corner. As an alternative to the NNMPCs, the control block also includes a state-of-the-art NMPC anti-jerk strategy, used as benchmark for the NNMPC implementations.
- The open-loop NN block, which generates the predicted system outputs, $\hat{y}_{k,i}$, used to assess the NN match with the plant, where the generic subscript k indicates a considered scalar variable within the respective vector, and the subscript i is the data sample index.
- The data buffer generator, storing a pre-defined number n of data samples of the relevant measured or estimated outputs from the vehicle, $y_{k,i}$, and those from the open-loop prediction model, $\hat{y}_{k,i}$.
- The online training and weight optimization block, which is another novel item of this research, managing the update of the controller by: i) opening and closing the parallel pool dedicated to the controller auto-training task, while the controlled vehicle simulation is running; ii) activating and deactivating the online training of the NN to be used as prediction model, and, once this is completed, the optimization of the NNMPC cost function weights with the retrained network. The process in ii) is kicked off when the root mean square errors (RMSE) between $y_{k,i}$ and $\hat{y}_{k,i}$, computed along n, exceed pre-defined thresholds, $RMSE_{th}$.
- The plant, i.e., the experimentally validated vehicle simulation model for control system assessment, developed in the Matlab/Simulink environment and parametrized according to [33].

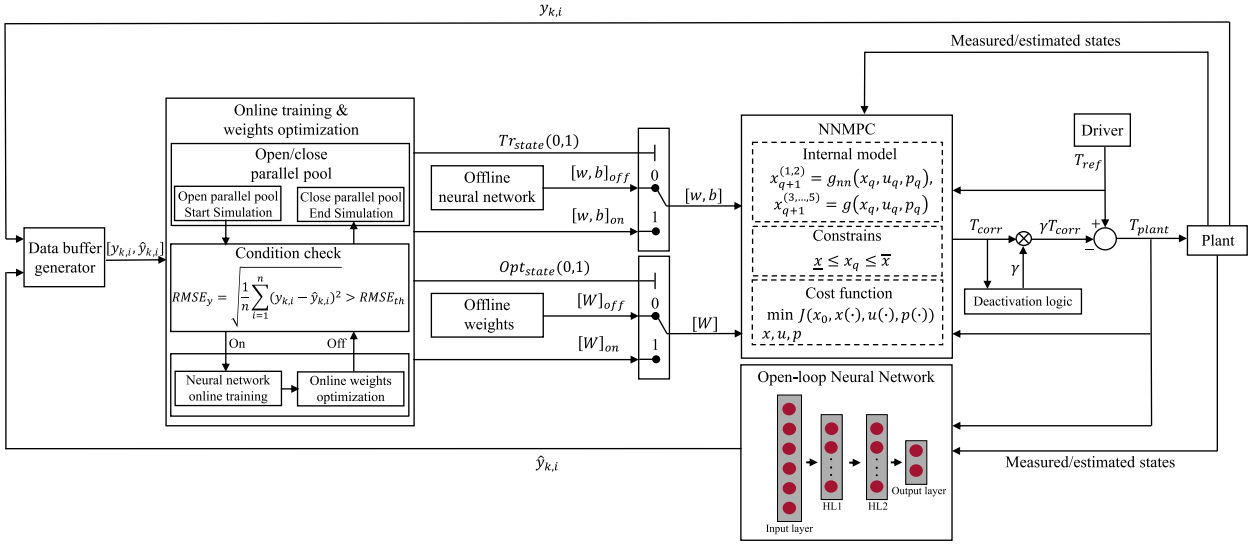


Fig. 1. Simplified schematic of the simulation and anti-jerk control architecture.

2.2. Case study electric vehicle and model validation

The case study EV is the Range Rover Evoque prototype that was implemented as the demonstrator of the European projects E-VECTOORC and iCOMPOSE [52–54], see Fig. 2(a). The vehicle is equipped with four on-board EMs, based on switched reluctance technology, each of them connected to its respective wheel through a single-speed transmission, constant-velocity joints, and a half-shaft, see Fig. 2(b). The main vehicle and drivetrain parameters are reported in Table 1.

The simulation model for control system assessment is the one from [33], which includes: i) the longitudinal vehicle body dynamics; ii) the EM torque dynamics, expressed through a transfer function; iii) the rotational drivetrain dynamics, with consideration of the torsional behaviour of the half-shafts and the equivalent drivetrain backlash (amounting to 2 deg at the wheels, in the nominal vehicle set-up), which is modelled through a piecewise formulation, with stiffness and damping contributions; iv) the dynamics of each driving wheel; v) the longitudinal load transfers; and vi) the nonlinear tyre slip behaviour, considered through version '94 of the Magic Formula model [55], with the variation of the relaxation length as a function of the longitudinal tyre slip, k_x, and vertical load, F_z, and a dedicated slip ratio formulation for ensuring tyre model stability in low-speed conditions [56,57]. Fig. 3 reports the longitudinal tyre force (F_x) characteristics of the considered tyres for different values of k_x and F_z, where Tyre 1 is the nominal tyre of the considered EV, while Tyre 2 is an alternative tyre, which – where explicitly indicated – has been used in the simulations to highlight the adaptability of the proposed anti-jerk controller.

The plant model has been validated through experimental tip-in manoeuvres carried out at the Lommel Proving Ground (Belgium), involving swift positive variations of T_{ref} (which is then kept constant), from different initial speeds and with multiple torque demands, see the examples in Fig. 4, indicating a good agreement between experiments and simulations. Hence, the nonlinear model can be considered a valid tool for control system assessment.

3. Controller formulations

3.1. Neural network prediction model

The NNMPC internal model, i.e., the digital twin used by the NNMPC for predicting the system response along the prediction horizon t_h, is implemented through a feedforward artificial neural network (FF-ANN), supported by physics-based formulations, see the schematic in Fig. 5.

The FF-ANN input vector, h, with $h \in \mathbb{R}^{n_h}$, where $n_h = 6$ is also the number of neurons in the input layer, is defined as:

$$h = \left[\dot{\vartheta}_1, \dot{\vartheta}_2, \Delta\dot{\vartheta}, \Delta\vartheta, T_{plant}, T_{em}\right]^T \tag{1}$$

where $\dot{\vartheta}_1$ is the rotational EM speed referred to the wheel, i.e., $\dot{\vartheta}_1 = \dot{\vartheta}_{em}/i_g$, with $\dot{\vartheta}_{em}$ being the angular speed of the EM rotor; $\dot{\vartheta}_2$ is the angular wheel speed; $\Delta\dot{\vartheta}$ and $\Delta\vartheta$ are the torsion rate and torsion angle of the half-shaft; T_{plant} is the reference EM torque sent to the inverter; and T_{em} is the actual EM torque, which is usually estimated by the inverter. All the components of h can be available – through appropriate sensors and estimators – in production EVs, to be used as initial conditions for the NNMPC prediction at each time step. The FF-ANN output vector, y, includes the motor acceleration at the wheel, and the wheel acceleration:

$$y = \left[\ddot{\vartheta}_1, \ddot{\vartheta}_2\right]^T \tag{2}$$

In the selected configuration, the FF-ANN consists of two feedforward hidden layers, each with $n_n = 16$ neurons. Such network architecture was the result of a trade-off analysis between prediction accuracy and computational efficiency of the resulting NNMPC, see the discussion in Section 5.6. Each output component of the first hidden layer, $a_{1,p}$, is given by:

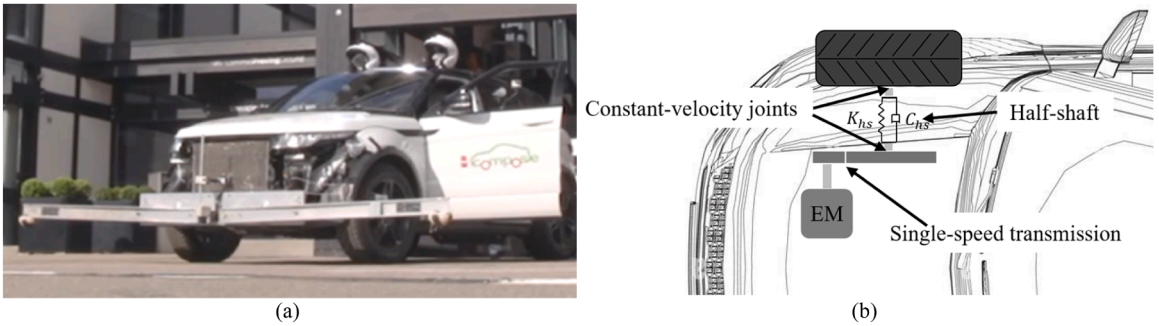


Fig. 2. (a) The considered EV prototype; and (b) Simplified layout of an individual drivetrain.

Table 1
Main vehicle and drivetrain parameters.

Parameter	Symbol	Value	Unit
Vehicle mass	M_v	2350	[kg]
Wheelbase	l	2.66	[m]
Aerodynamic drag coefficient	C_d	0.33	[-]
Frontal vehicle area	A_v	2.2	[m^2]
Peak power of the individual electric powertrain	$P_{em,max}$	88	[kW]
Transmission gear ratio	i_g	10.5	[-]
Torsion stiffness of the half-shafts	K_{hs}	7000	[Nm/rad]
Wheel mass moment of inertia	J_w	1.5	[kgm^2]
Wheel radius	R_w	0.37	[m]

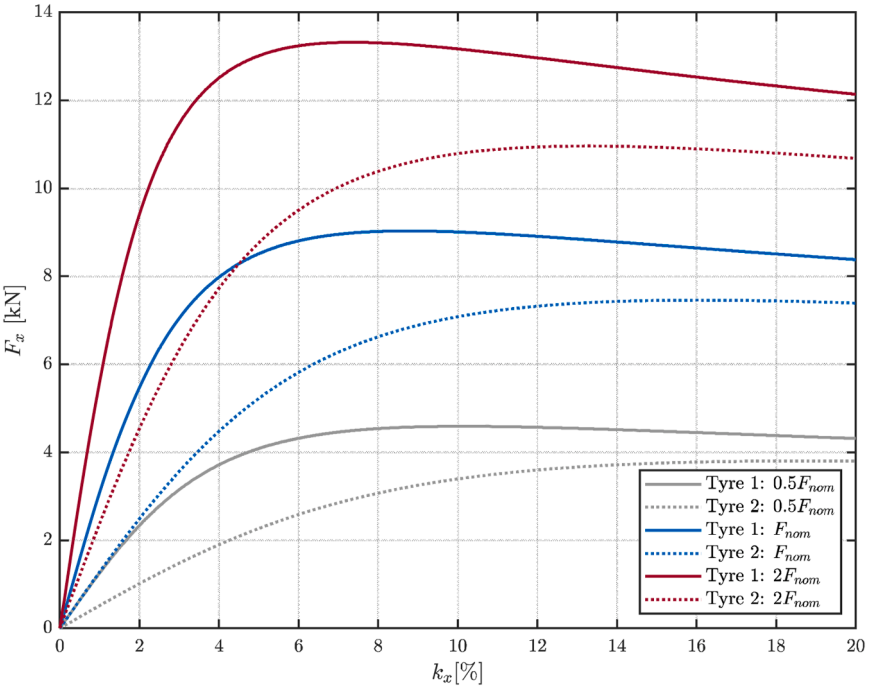


Fig. 3. Longitudinal tyre force characteristic as a function of the slip ratio, for three vertical tyre loads – expressed as a function of a nominal load value $F_{nom} = M_v g/4$ – and two tyre model parametrisations (Tyre 1, based on the model parameters from the European projects E-VECTOORC and iCOMPOSE, see also [33], and Tyre 2, based on parameters from the European project EVC1000, see [17]).

$$\begin{cases} y_{1,p} = \sum_{j=1}^{n_h} w_{1,pj} h_j + b_{1,p} \\ a_{1,p} = f_1\left(y_{1,p}\right) \end{cases} \tag{3}$$

where the index p, with $p = 1,..,16$, refers to the considered neuron; $w_{1,pj}$ is the weight connecting the input h_j, with $j = 1,.., 6$, to the p-th neuron in the first hidden layer; $b_{1,p}$ is the bias associated with the p-th neuron in the first layer; and f_1 is the activation function applied to the output of each neuron. In vectorial form, (3) can be expressed as:

$$\begin{cases} y_1 = W_1^T h + b_1 \\ a_1 = f_1(y_1) \end{cases} \tag{4}$$

where y_1 is the output vector of the first hidden layer; $W_1 \in \mathbb{R}^{n_h \times n_n}$ is the weight matrix of the same layer; $b_1 \in \mathbb{R}^{n_n}$ is the bias vector; and $a_1 \in \mathbb{R}^{n_n}$ is the vector of the activated outputs from the first hidden layer. Similarly, the vectorial input/output formulation for the second hidden layer is:

$$\begin{cases} y_2 = W_2^T a_1 + b_2 \\ a_2 = f_2(y_2) \end{cases} \tag{5}$$

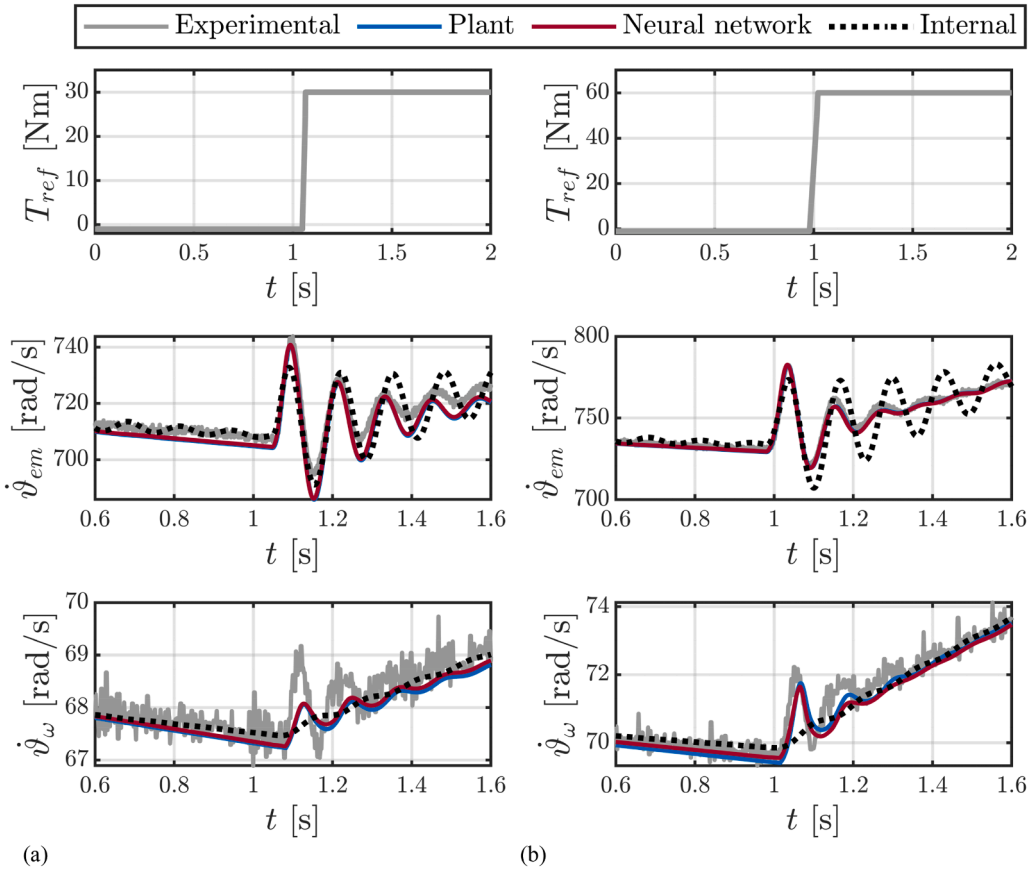


Fig. 4. Model validation along tip-in tests, with final (a) 30 Nm and (b) 60 Nm T_{ref} values, from an initial EV speed of ~90 km/h. Comparison of experimental results ('Experimental'), and simulation results from the model for control system assessment ('Plant'), the nominal NN-based model ('Neural network') generating the NNMPC prediction, and the physics-based model ('Internal') providing the NMPC prediction.

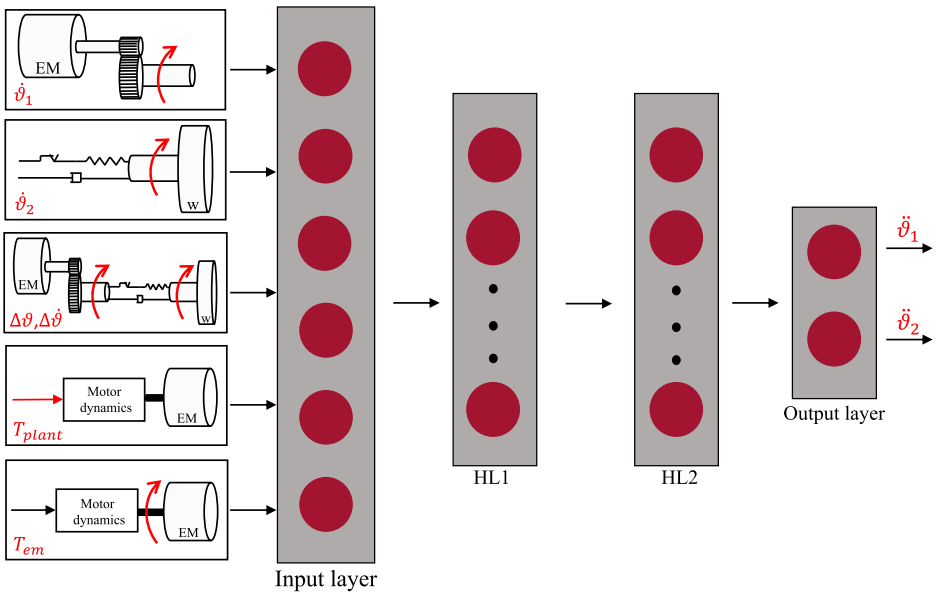


Fig. 5. Schematic of the FF-ANN architecture for the prediction model.

where y_2, W_2, b_2, f_2, and a_2 are the respective output vector, weight matrix, bias vector, activation function, and vector of the activated outputs. Finally, the FF-ANN output is given by:

$$y = W_3^T a_2 + b_3 \tag{6}$$

where W_3 and b_3 are the weight matrix and bias vector of the output layer.

In summary, (1)-(6) can be re-arranged as:

$$\ddot{\vartheta}_1 = \sum_{l=1}^{n_n} w_{3,\ddot{\vartheta}_1,l} \left\{ f_2 \left[\sum_{m=1}^{n_n} w_{2,m} f_1 \left(\sum_{j=1}^{n_h} w_{1,j} h_j + b_1 \right) + b_2 \right] \right\} + b_{3,\ddot{\vartheta}_1} \tag{7}$$

$$\ddot{\vartheta}_2 = \sum_{l=1}^{n_n} w_{3,\ddot{\vartheta}_2,l} \left\{ f_2 \left[\sum_{m=1}^{n_n} w_{2,m} f_1 \left(\sum_{j=1}^{n_h} w_{1,j} h_j + b_1 \right) + b_2 \right] \right\} + b_{3,\ddot{\vartheta}_2} \tag{8}$$

where $w_{1,j}$ and $w_{2,m}$ are the relevant weight vectors within W_1 and W_2; $w_{3,\ddot{\vartheta}_1,l}$ and $w_{3,\ddot{\vartheta}_2,l}$ are the weights of the output layer; and $b_{3,\ddot{\vartheta}_1}$ and $b_{3,\ddot{\vartheta}_2}$ are the corresponding biases.

In the prediction model, $\dot{\theta}_1$ and $\dot{\theta}_2$ are computed through the numerical integration of the respective accelerations along time; $\Delta\dot{\vartheta}$ is obtained from its kinematic definition, i.e., $\Delta\dot{\vartheta} = \dot{\vartheta}_1 - \dot{\vartheta}_2$, which, through integration along time, brings $\Delta\vartheta$.

In parallel to the NN, the inverter and EM dynamics are considered through a first order formulation:

$$T_{em} = T_{plant} - \tau_{em}\dot{T}_{em} \tag{9}$$

where τ_{em} is the torque time constant of the powertrain, and T_{plant} in (9) is calculated as:

$$T_{plant} = T_{ref} - T_{corr} \tag{10}$$

The integration of $\dot{T}_{em}$ enables the computation of all the inputs to the FF-ANN, which can thus be used as core component of the

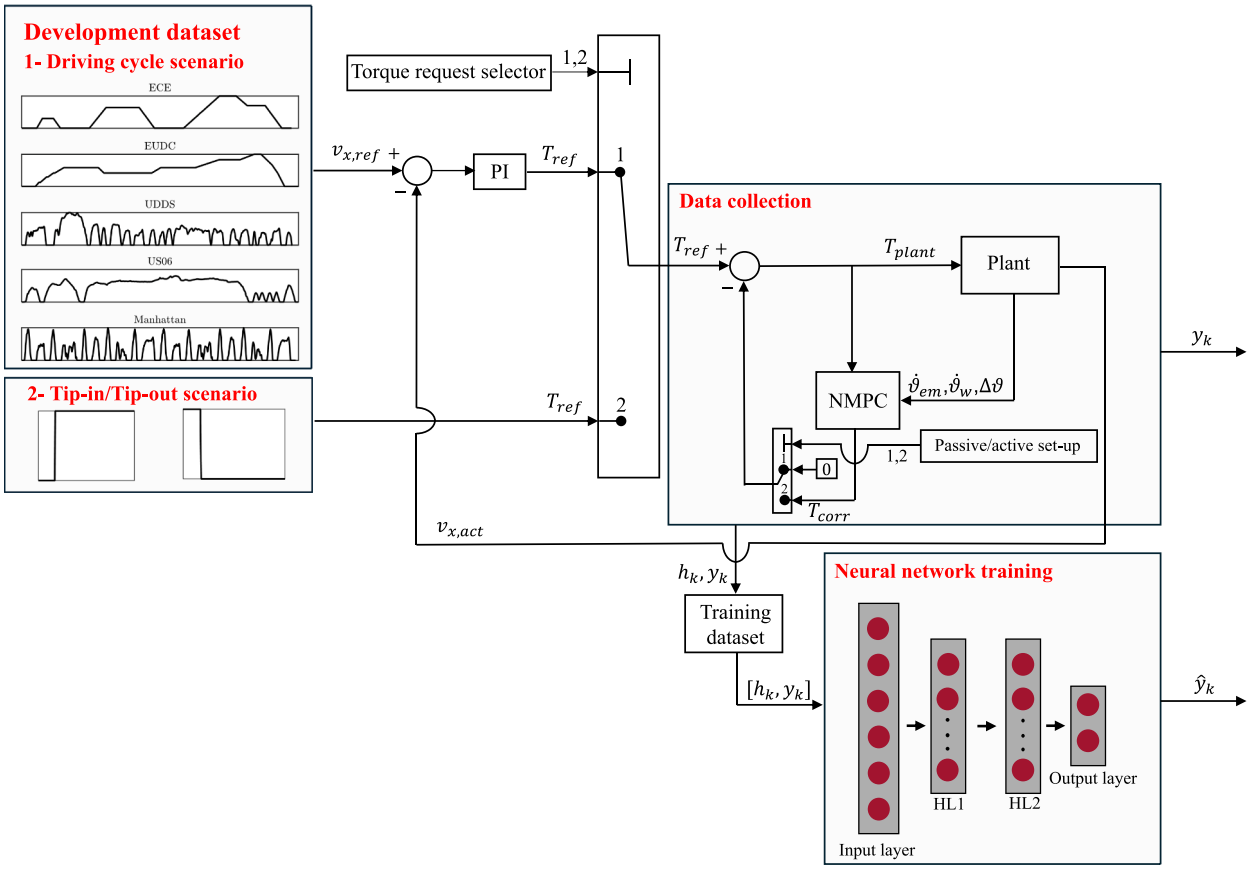


Fig. 6. Schematic of the simulation architecture for the offline training process.

NNMPC prediction.

3.2. Offline training of the neural network

The initial training of the FF-ANN, based on data from the nominal plant, is referred to as offline as it is carried out in the controller design stage, outside the online simulation environment in Section 2.1, although the plant model was used to generate the training data, covering both the front-wheel-drive and four-wheel-drive operation of the considered EV, in its passive (i.e., without anti-jerk control) and active (i.e., with the baseline NMPC-based anti-jerk controller in Section 3.5) set-ups. The NN development dataset, see Fig. 6, was generated by simulating multiple scenarios: i) tip-in tests from different initial speeds, as well as several initial/final torque demand values, torque gradients, and time instants for the torque demand increase; ii) tip-out tests, with negative swift torque demand variations, with similar variety of conditions as for i); and iii) standardised driving cycles [58]. The resulting dataset was divided into training, validation, and testing sets, corresponding to 85%, 10%, and 5% of the total data.

The loss function for network training is the mean squared error (MSE), computed across the two relevant accelerations $\ddot{\vartheta}_1$ and $\ddot{\vartheta}_2$, which reflects the fitting accuracy between the NN model and training data:

$$MSE = \frac{1}{n}\sum_{k=1}^{2}\sum_{i=1}^{n_d}\left[y_{k,i} - \widehat{y}_{k,i}\right]^2 \tag{11}$$

where n_d is the total number of values for the considered dataset. During the offline FF-ANN design phase, three activation functions were evaluated, see Fig. 7(a), i.e., the hyperbolic tangent ('Tanh' in the plot), rectified linear unit ('ReLU'), and swish functions [59]. For emulating the output data, i.e., the angular accelerations, which can take both positive and negative values in the context of the specific highly nonlinear system, the swish function provided superior performance, and was selected for the NNMPC implementation. Hence, $f(y)$ in (4) and (5) are expressed as:

$$f(y) = y\sigma(y) = y\frac{1}{1+e^{-y}} \tag{12}$$

where $\sigma(y)$ is the sigmoid function. The offline training process was carried out through the Deep Learning Toolbox of Matlab, with the hyperparameter settings in Table 2. Fig. 7(b) is an example of training process results, expressed in terms of loss over the training and validation datasets. Because of the large size of the training dataset, the validation loss is already stabilized by the second training iteration. As shown in Fig. 4, the resulting FF-ANN model closely replicates the experimental EV behaviour, with similar level of match as the nonlinear model for control system assessment (i.e., the 'Plant' in the plot), and outperforms the simplified physics-based model of the benchmarking NMPC algorithm.

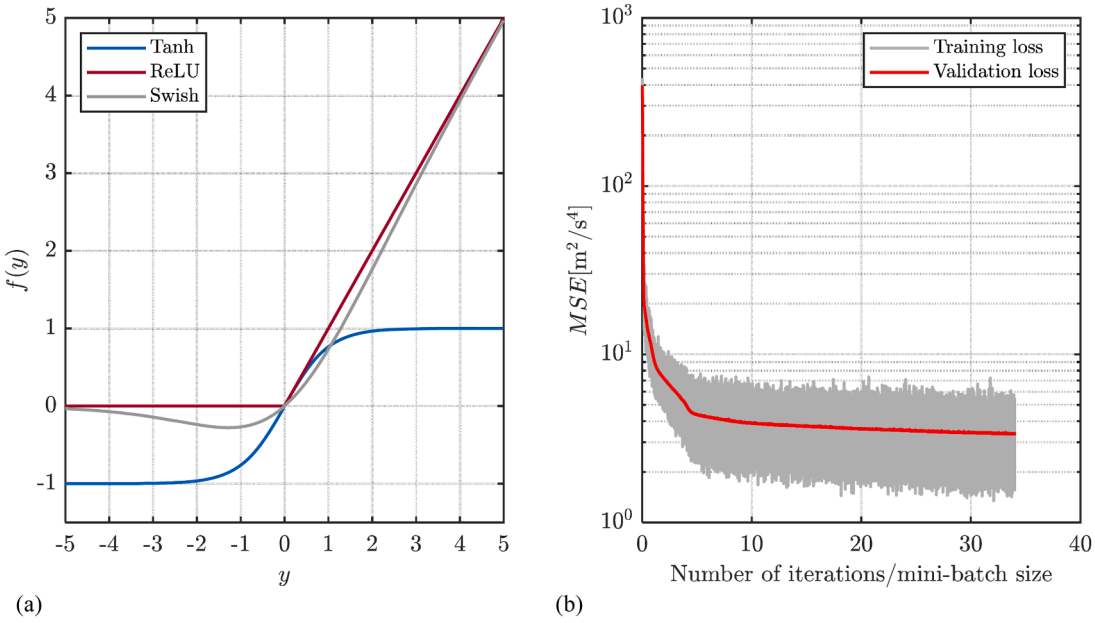


Fig. 7. (a): Shape of the tested activation functions; and (b): Example of MSE trends resulting from the neural network training and validation process.

Table 2
Neural network hyperparameter settings (offline training).

Hyperparameter group	Hyperparameter	Value
Neural network architecture	Hidden layers	2
	Neurons in first layer	16
	Neurons in second layer	16
	Activation function	Swish
Training process	Mini batch size	13000
	Initial learning rate	0.001
	Maximum epochs	500
	Optimization algorithm	Adam
	Loss function	MSE

3.3. Nonlinear optimal control problem formulation

At each iteration, the NNMPC algorithm computes the online solution of the following OCP, to minimize a cost function J subject to (s.t.) an appropriate set of constraints:

$$J = \min_{x(\cdot),u(\cdot),p(\cdot)} \sum_{q=0}^{N-1} \left[\| x_q - x_{q,ref} \|_Q^2 + \| u_q - u_{q,ref} \|_R^2 \right] + \| x_N - x_{N,ref} \|_{Q_N}^2 \tag{13}$$

$$s.t. \tag{13a}$$

$$x_0 = x_c \tag{13b}$$

$$x_{q+1}^{(1,2)} = g_{nn}\left(x_q, u_q, p_q\right) \tag{13c}$$

$$x_{q+1}^{(3,\ldots,5)} = g\left(x_q, u_q, p_q\right) \tag{13d}$$

$$\underline{x} \leq x_q \leq \overline{x} \tag{13e}$$

where $x = [\dot{\vartheta}_1, \dot{\vartheta}_2, \Delta\vartheta, \Delta\dot{\vartheta}, T_{em}]^T$ is the state vector; $u = [T_{corr}]$ is the control action vector; $p = \left[T_{ref}, T_{plant}\right]^T$ is the parameter vector; x_c is the current value of the state vector, i.e., at the beginning of the prediction; the subscript q refers to a generic prediction step; N is the number of prediction steps, which is coincident with the number of control steps; g_{nn} and g refer to the FF-ANN-based and physics-based components of the prediction model in (1)-(10); the notations $x^{(1,2)}$ and $x^{(3,\ldots,5)}$ indicate the relevant components of x; $\underline{x}$ and $\overline{x}$ are the lower and upper bounds on the states, formulated as hard constraints on the EM torque; and $Q \in \mathbb{R}^{5\times5}$, $R \in \mathbb{R}^{1\times1}$, and $Q_N \in \mathbb{R}^{5\times5}$ are positive diagonal weight matrices:

$$Q = \mathrm{diag}([0, 0, 0, W_{\Delta\dot{\vartheta}}, W_{T_{em}}]) \tag{14}$$

Algorithm 1
Activation and deactivation algorithm of the anti-jerk controller.

Initialize the activation gain γ, the anti-jerk corrective torque threshold $T_{corr,tr}$, the anti-jerk corrective torque rate threshold $\dot{T}_{corr,tr}$, and the time counter threshold $t_{count,tr}$.

```
if (|T_corr| ≥ T_corr,tr) || (|Ṫ_corr| ≥ Ṫ_corr,tr)
    γ = 1
    t_count = 0
    while t_count ≤ t_count,tr
        γ = 1
        t_count = t_count + Δt
        if (|T_corr| ≥ T_corr,tr) || (|Ṫ_corr| ≥ Ṫ_corr,tr)
            t_count = 0
        end
    end
    while (t_count ≤ t_count,tr + Δt_lin) && (|T_corr| < T_corr,tr) && (|Ṫ_corr| < Ṫ_corr,tr)
        γ = 1 − [1 / Δt_lin][t_count − t_count,tr]
        t_count = t_count + Δt
    end
else
    γ = 0
    t_count = 0
end
```

$$R = W_{T_{corr}} \tag{15}$$

$$Q_N = Q \tag{16}$$

Based on (13)-(16), J incorporates a term targeting the reduction of the drivetrain oscillations through the minimization of $\Delta\dot{\vartheta}$; a term related to the tracking of the driver torque demand, T_{ref}; and a term to reduce the control effort expressed by T_{corr}.

3.4. Controller activation and deactivation

Similarly to other anti-jerk control implementations from the literature [33], the NNMPC algorithm is activated only when this is beneficial to the vehicle response. The activation and deactivation are managed through a variable gain γ, with $0 \leq \gamma \leq 1$, which is multiplied by T_{corr}. According to the pseudo-code in Algorithm 1 and Fig. 8, the anti-jerk function is activated when the magnitude of its corrective torque and/or the one of the respective time derivative are larger than defined thresholds, referred to as $T_{corr,tr}$ and $\dot{T}_{corr,tr}$. Vice versa, if the magnitudes of T_{corr} and its time derivative are both lower than the same thresholds, a counter is activated, which outputs the time t_{count} in which the anti-jerk control input remains low. Once t_{count} exceeds the threshold $t_{count,tr}$, if the low-intensity condition of the anti-jerk intervention persists, γ is linearly reduced and saturated to zero, condition in which the control function is deactivated. In the pseudo-code, Δt refers to the controller implementation step, while Δt_{lin} is the time taken by γ to transition from 1 to 0.

3.5. Benchmarking nonlinear model predictive controller

The internal model formulation of the benchmarking NMPC anti-jerk algorithm is based on the one in [33] and [56], which consider a model with two mass moments of inertia, the first one, J_1, representing the rotating parts of the EM and drivetrain, and the second one, J_2, corresponding to the equivalent inertia of the wheel and the vehicle. J_1 and J_2 are connected by a torsional spring with a 2α backlash magnitude, and a torsional damper, see Fig. 9, where $K_{hs,s}$ and $C_{hs,s}$ indicate the respective stiffness and damping coefficient. Such model, which is shown to provide good anti-jerk control performance in several studies [60–63], is described, on top of the first order EM dynamics formulation in (9), by the following torque balance equations:

$$\ddot{\vartheta}_1 = \frac{1}{J_1}\left[\eta_g T_{em} i_g - T_{hs}\right] \tag{17}$$

$$\ddot{\vartheta}_2 = \frac{1}{J_2}\left[T_{hs} - 0.5T_{aero} - 0.5T_{roll}\right] \tag{18}$$

where η_g is the gearbox efficiency; and T_{hs} is the half-shaft torque, which is given by:

$$T_{hs} = 0.5K_{hs,s}[\Delta\vartheta - \alpha]\{\tanh(K_1[\Delta\vartheta - K_2]) + 1\} + 0.5K_{hs,s}(\Delta\vartheta + \alpha)\{\tanh[-K_1[\Delta\vartheta + K_2]] + 1\} + C_{hs,s}\Delta\dot{\vartheta} \tag{19}$$

where K_1 and K_2 are tuning parameters, which approximate the piecewise linear behaviour corresponding to the mechanical play. In

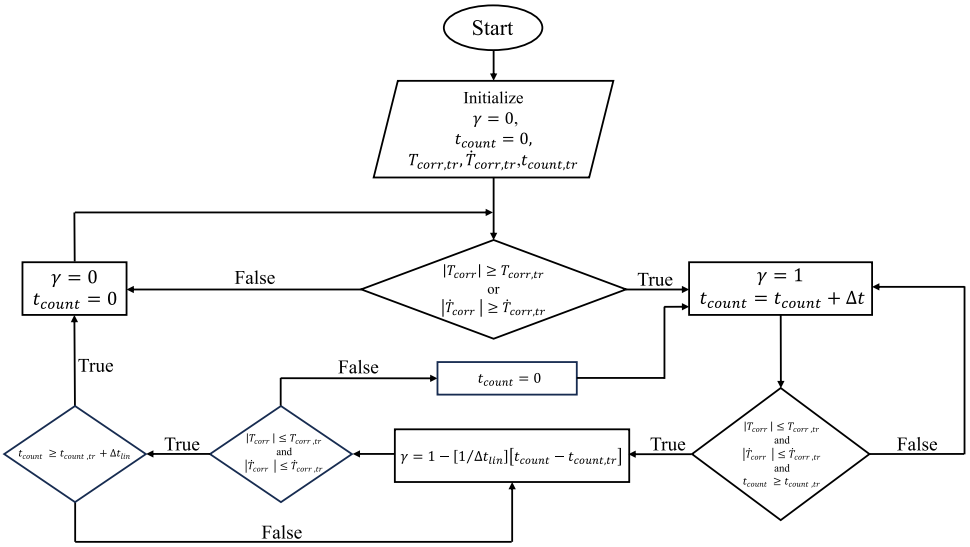


Fig. 8. Simplified flow chart of the activation and deactivation algorithm of the anti-jerk controller.

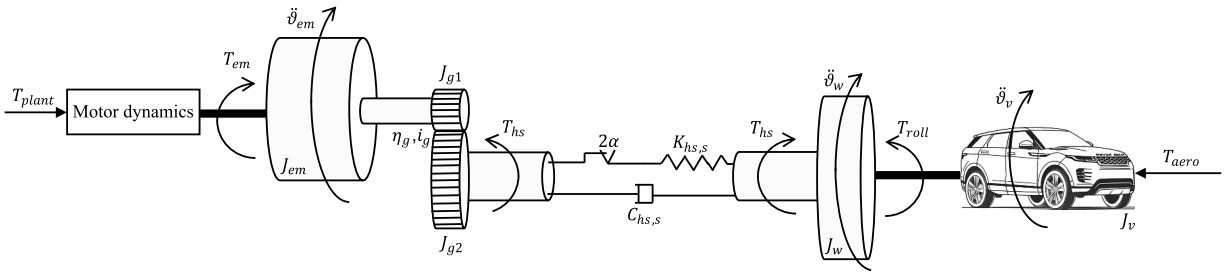


Fig. 9. Schematic of the internal model of the benchmarking NMPC strategy.

(18), T_{aero} and T_{roll} are the equivalent moments caused by the aerodynamic drag and rolling resistance effects:

$$T_{aero} = 0.5 R_w \rho A_v C_x v^2 \tag{20}$$

$$T_{roll} = R_w \left[f_0 M_v g + f_1 M_v g v_x^2 \right] \tag{21}$$

where ρ is the air density; g is the gravitational acceleration; and f_0 and f_1 are the rolling resistance coefficients.

Given the significant simplification of the system dynamics intrinsic to the model, which, for example, neglects the damping effect of the longitudinal tyre slip, $K_{hs,s}$ and $C_{hs,s}$ were selected to provide a reasonable match with the plant model for control system assessment and the experiments, see the open-loop model comparison in Fig. 4, rather than being chosen directly from the half-shaft properties. Nevertheless, while such simplified physics-based prediction model provides similar level of accuracy to the FF-ANN of the NNMPC in terms of motor speed emulation, it is by far inferior to the network in terms of wheel speed dynamics. The OCP of the NMPC algorithm is the same as for the NNMPC, see (13)-(16).

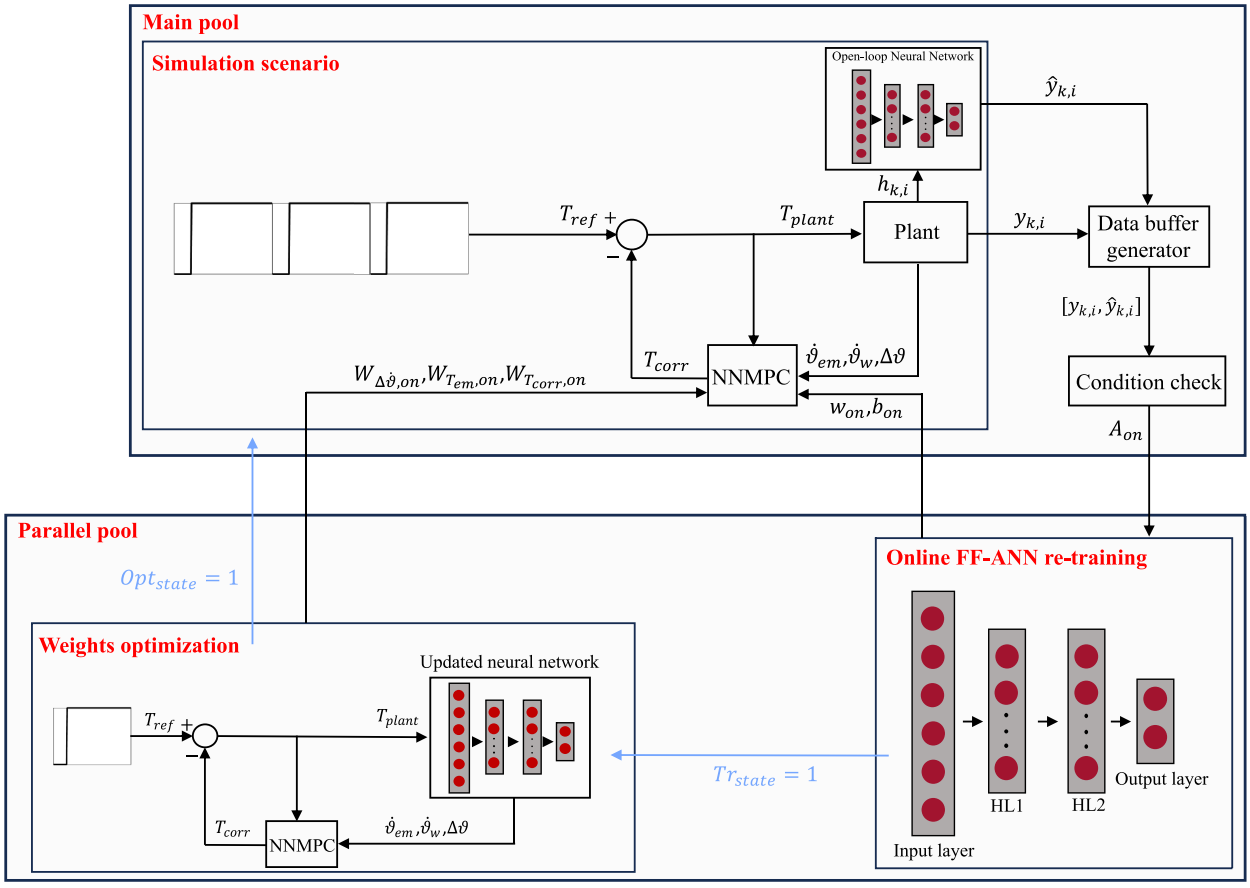


Fig. 10. Simplified schematic of the adaptation architecture.

3.6. Controller implementation

The acados toolkit [64], providing a robust platform for embedded optimizations, is adopted for the online solution of the nonlinear OCPs associated with the proposed NMPC and NNMPC implementations. The solver was configured to use the sequential quadratic programming method, with the partial condensing option, and the fourth order implicit Runge Kutta integrator. For simplicity, unless otherwise specified, in the proof-of-concept simulations of this study, similarly to the set-up in [33], the controller sampling time T_s and the number N of prediction steps are set to 1 ms and 4, while the maximum number of iterations is 4. The implications of the main control parametrisations will be analysed in the following Section 5.7.

4. On-line controller adaptation

The online adaptation of both the NN prediction model and NNMPC cost function weights is managed through a Stateflow chart in Matlab-Simulink. The dual adaptation process enables the controller self-calibration, essential for maintaining optimal performance under operational conditions not considered during the offline training process. The building blocks, outlined in Fig. 1 and detailed in Fig. 10 and the pseudocode of Algorithm 2, are:

- The main pool, which covers: i) the online operation of the controller, with the latest version of the prediction model calibration, which is kept constant between subsequent adaptations. Hence, the pool includes both the predictive anti-jerk controller and the high-fidelity simulation model of the plant, where the latter would be absent in case of experimental vehicle deployment; and ii) the condition check block, which determines whether a prediction model update must be initiated. In this respect, for the current calibration of the FF-ANN and control weights, the condition check periodically computes the root mean squared error, $RMSE_y$, between the relevant angular accelerations from the plant, obtained from the sensor measurements, $y_{k,i}$, and those, $\widehat{y}_{k,i}$, predicted by the FF-ANN calibration currently used by the predictive controller, which is run in open-loop outside the controller (see the 'Open-loop Neural Network' block in Fig. 10):

$$RMSE_y = \sqrt{\frac{1}{n}\sum_{k=1}^{2}\sum_{i=1}^{n_{d,ol}}\left[y_{k,i} - \widehat{y}_{k,i}\right]^2} \tag{22}$$

where $n_{d,ol}$ is the number of data points for the $RMSE_y$ computation. The $y_{k,i}$ terms are the time derivatives of the measurements of the EM and wheel speeds, in the specific application coming from the simulation model of the plant. Once $RMSE_y$ exceeds a threshold $RMSE_{th}$, a flag variable is generated to start the parallel update of the network, and the relevant buffer of sampled data, A_{on}, is sent to the parallel pool for online training. A_{on} has the following matrix form:

Algorithm 2
Simplified online re-training of the FF-ANN, and NNMPC cost function weights optimization.

Initialise the online variables:
  $\left[W_{(1,2,3)}, b_{(1,2,3)}, W_{(\Delta\dot{\vartheta},T_{em},T_{corr})}\right] = \left[W_{(1,2,3),off}, b_{(1,2,3),off}, W_{(\Delta\dot{\vartheta},T_{em},T_{corr}),off}\right]$;
  Initialise: $RMSE_y = 0$; $Tr_{state} = 0$; $Opt_{state} = 0$;
Open parallel pool; Start simulation scenario
if $t/\Delta t$ is integer
  Evaluate $RMSE_y$
  if $RMSE_y < RMSE_{th}$
    $Tr_{state} = 0$; $Opt_{state} = 0$;
  elseif $RMSE_y \geq RMSE_{th}$
Start neural network online training in a parallel pool
      if $Tr_{state} == 0$
       $Opt_{state} = 0$
       $\left[W_{(1,2,3)}, b_{(1,2,3)}\right] = \left[W_{(1,2,3),off}, b_{(1,2,3),off}\right]$
      elseif $Tr_{state} == 1$
        $\left[W_{(1,2,3)}, b_{(1,2,3)}\right] = \left[W_{(1,2,3),on}, b_{(1,2,3),on}\right]$
Start NNMPC cost function weights optimization in the same parallel pool
      end
  if $Tr_{state} == 1$ && $Opt_{state} == 0$
    $\left[W_{(\Delta\dot{\vartheta},T_{em},T_{corr})}\right] = \left[W_{(\Delta\dot{\vartheta},T_{em},T_{corr}),off}\right]$
  elseif $Tr_{state} == 1$ && $Opt_{state} == 1$
$\left[W_{(\Delta\dot{\vartheta},T_{em},T_{corr})}\right] = \left[W_{(\Delta\dot{\vartheta},T_{em},T_{corr}),on}\right]$
  end
  end
end
End simulation scenario; close parallel pool

$$A_{on} = \begin{bmatrix} h_{1,on} & h_{2,on} & \cdots & h_{n,on} \\ y_{1,on} & y_{2,on} & \cdots & y_{n,on} \end{bmatrix}^T \tag{23}$$

where the $h_{i,on}$ components are the neural network input vectors according to (1), which are collected along an appropriate time interval Δt; and $y_{i,on}$ are the target FF-ANN output vectors. The buffer dataset size, n, depends on the time duration Δt corresponding to the buffer itself, and the time step Δt_s between each sample:

$$n = \Delta t / \Delta t_s \tag{24}$$

- A parallel pool w.r.t. the simulation scenario running in the main pool, for FF-ANN re-training followed by the NNMPC cost function weight adaptation, based on real-time operational data. The parallel pool operates through the parallel computing toolbox of Matlab, and includes:

  i. The online FF-ANN re-training block, which adjusts all the FF-ANN weights and biases (W_1, W_2, W_3, b_1, b_2, and b_3) to reflect the updated plant dynamics, where the training process is conducted and completed based on the loss function in (11), see Section 3.2. The mini batch size, initial learning rate, and maximum number of training epochs, see Table 3, are selected to perform 100 iterations at each epoch, and accelerate the training process w.r.t the offline training case. The training status is indicated by the flag variable Tr_{state}, which is 0 during online training, and 1 at the training completion. Tr_{state} is used to activate the NNMPC cost function weight optimization, see Fig. 10. Hence, the neural network model becomes a digital twin of the system, receiving data from the real plant, and transmitting its updated parametrisation to the control system.
  ii. The NNMPC cost function weight optimization block, where the updated FF-ANN serves as an updated plant model for optimizing $W_{\Delta\dot{\vartheta}}$, $W_{T_{em}}$, and $W_{T_{corr}}$ of (14) and (15). The FF-ANN is used to this purpose, since during real-world controller operation the high-fidelity model of the main pool is obviously not available. The optimization is carried out for a representative set of tip-in tests, and is based on a dedicated auxiliary cost function J_{opt}, different from the one of the model predictive controller to directly consider the relevant indicators in their typical form, and an iterative numerical routine using the *surrogateopt* algorithm, according to the following formulation:

$$J_{opt} = \min_{W_{NNMPC}} \left(W_{opt,1} VDV_{a_x^*} + W_{opt,2} RMS_{a_x^*} + W_{opt,3} \Delta v_x + W_{opt,4} IACA + W_{opt,5} \Delta_{t,a_x} \right) \tag{25}$$

$$s.t.$$

$$W_{NNMPC,min} \leq W_{NNMPC} \leq W_{NNMPC,max} \tag{25b}$$

where W_{NNMPC} is the vector of the NNMPC cost function weights; $W_{NNMPC,min}$ and $W_{NNMPC,max}$ are the vectors including the weight bounds; and the terms in (25), defined according to [33] and [65,66], include:

a. The fourth power vibration dose value, $VDV_{a_x^*}$, which is related to the resulting comfort level:

$$VDV_{a_x^*} = \sqrt[4]{\int_{T_1}^{T_2} a_x^{*4} dt} \tag{26}$$

where a_x^* is the zero-mean value of a_x, obtained by high-pass filtering the longitudinal acceleration; and T_1 and T_2 are the initial and final time instants of the relevant section of the manoeuvre.

b. The root-mean square value of a_x^*, which is another comfort indicator:

Table 3
Parameters of the online FF-ANN re-training process.

Hyperparameter	Value
Mini batch size	$n/100$
Initial learning rate	0.005
Maximum number of epochs	400
Optimization algorithm	Adam
Loss function	MSE

$$RMS_{a_x^*} = \sqrt{\frac{1}{T_2 - T_1} \int_{T_1}^{T_2} a_x^{*2} dt} \tag{27}$$

c. The anti-jerk control induced reduction, Δv_x, of the longitudinal vehicle speed:

$$\Delta v_x = v_{x,passive}(T_2) - v_{x,active}(T_2) \tag{28}$$

where $v_{x,passive}(T_2)$ and $v_{x,active}(T_2)$ are the longitudinal speed values of the passive and controlled vehicles at the end of the manoeuvre.

d. The time delay, Δ_{t,a_x}, between the application of a step input in terms of EM torque request, and the subsequent achievement of a longitudinal vehicle acceleration level, $a_{x,ref}$, arbitrarily set to 50% of the steady-state a_x value for the passive vehicle. Δ_{t,a_x} is a vehicle responsiveness indicator.

e. The integral of the absolute value of the control action, *IACA*, averaged with the relevant manoeuvre duration, which is a control effort indicator:

$$IACA = \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} |T_{corr}| dt \tag{29}$$

The flag variable Opt_{state} in Fig. 10 is 0 during the update process of the NN and control weights, and becomes 1 at the completion of the weight optimization. At the end of the process, within the NNMPC algorithm running in the main pool, the current NN parameters and weights are replaced with the respective updated values, and the routine is repeated every time $RMSE_y$ exceeds $RMSE_{th}$.

5. Preliminary proof-of-concept results

5.1. Considered simulation use cases

To preliminarily assess the potential of the proposed proof-of-concept self-adaptive control architecture, the anti-jerk algorithms are evaluated along the following use cases, which are simulated with the experimentally validated model in Section 2.2:

- Use case 1: performance comparison of the NNMPC, NMPC, and passive configurations, operating on the nominal plant, along a tip-in test with T_{ref} raising from -3 to 60 Nm in 10 ms, from an initial vehicle speed of 30 km/h. In this analysis, the FF-ANN of the NNMPC implementation has already been trained for a comprehensive set of manoeuvres, and does not need any adaptation, while the controller cost function weights have been optimised offline, with a routine similar to the one in Section 4.
- Use case 2: analysis of the NNMPC adaptation capability to different operating conditions, represented by a tip-in from -3 to 120 Nm, w.r.t. those in which the internal model has been initially trained offline, i.e., the nominal tip-in of use case 1, followed by a tip-out from 60 to -3 Nm, from approximately the same initial speed. Although such a very limited offline training – different from the comprehensive one of use case 1 – is sufficient for desirable NNMPC performance along the two considered manoeuvres used in the network training phase, the initial calibration of the algorithm cannot provide desirable results in the new testing conditions, which – instead – can be addressed through FF-ANN re-training. In this use case, to evaluate the potential benefit of the NNMPC adaptation, the update is firstly implemented offline, i.e., by retraining the FF-ANN and optimizing the weights separately from the simulations, without the accelerated re-training parametrization in Table 3 w.r.t. the one in Table 2. The offline update is followed by simulations of the NNMPC using the initial and updated calibrations. Secondly, for preliminarily demonstrating the implementation of the online update, i.e., during controller operation, use case 2 includes a manoeuvre consisting of a sequence of alternated tip-ins and tip-outs to/from the new torque value, i.e., 120 Nm. During the ongoing simulation, the parallel pool executes the concurrent online re-training of the network and the control weight optimization, according to the routine in Section 4. Once the parallel pool has transferred the new calibration parameters to the online NNMPC algorithm, the control system performance varies accordingly. At this stage, the tip-in simulation from -3 to 120 Nm is repeated, to compare the adaptation benefit brought by the online update w.r.t. the one of the offline routine, which should be the set-up providing the ideal update results.
- Use case 3: assessment of the NNMPC adaptation capability to varying vehicle parameters. The initial controller is based on an offline network calibration for the nominal EV configuration, along a portion (i.e., the initial 190 s) of the urban dynamometer driving schedule (UDDS, [58]). The assessment is carried out along the same driving cycle, while individually varying the following EV parameters: i) driveline backlash, i.e., α is incremented from 1 (nominal backlash parameter value referred to the wheel) to 2 deg; ii) EM torque time constant, i.e., τ_{em} is increased from 2.2 to 30 ms; and iii) tyre parameters, i.e., the Pacejka tyre model parametrisation is switched from the nominal Tyre 1 to Tyre 2 in Fig. 3. This use case directly implements the online NNMPC

adaptation in the initial phase of the driving cycle simulation. The UDDS simulation for the updated controller is then repeated at the end of the process, to evaluate the resulting performance.

- Use case 4: evaluation of the controller auto-tuning capability for different vehicle application and operating conditions. In this case, the initial FF-ANN has received the same very limited training as in use case 2, i.e., the offline network training has been carried out for the same tip-in / tip-out to / from 60 Nm, and the nominal EV plant (a D-segment sports utility vehicle, SUV) in Section 2.2. The NNMPC is then simulated on a higher category EV (an E-segment SUV), with different geometric, inertial, aerodynamic, tyre, powertrain and driveline parameters, and the online adaptation process is implemented along the sequence of tip-ins and tip-outs of use case 2. At the end of the process, the -3 to 120 Nm tip-in from 30 km/h is carried out again, to highlight the adaptation benefits.

It must be pointed out that the proposed anti-jerk control architecture is based on the update of the same neural network throughout the vehicle life time, which is initiated based on the magnitude of the appropriately selected error variables, see Section 4. The adoption of different initial trainings of the neural network within the NNMPC algorithms of the considered use cases was only implemented with the purpose of highlighting the potential benefit of the proposed adaptation architecture through a limited set of preliminary simulations, and ensuring ease of readability.

Some of the following analyses also include the comparison of the proposed model predictive anti-jerk algorithms with a benchmarking tachometric anti-jerk controller, according to the formulation in [33], which is beyond – in terms of complexity and performance – the implementations normally adopted in production passenger cars.

5.2. Use case 1: controller comparison in nominal conditions

Fig. 11 reports the time profiles of the actual EM torque, T_{em}, and longitudinal vehicle acceleration, a_x, for the passive EV, and the same EV controlled by the proposed NNMPC and the benchmarking NMPC and tachometric algorithms, while Table 4 includes the relevant KPIs, as well as the J_{opt} values. All controllers enhance comfort through the reduction of the longitudinal acceleration oscillations, while limiting the T_{ref} correction. W.r.t. the passive vehicle, NNMPC reduces $VDV_{a_x^*}$ and $RMS_{a_x^*}$ by 41% and 60%. More importantly, compared to the NMPC case, the NNMPC algorithm brings 4% and 9% $VDV_{a_x^*}$ and $RMS_{a_x^*}$ reductions. Also, NNMPC reduces J_{opt} by 45% w.r.t. the passive vehicle, and 4% and 6% w.r.t the NMPC and tachometric implementations. The better performance of the NNMPC implementation is caused by the more accurate prediction model w.r.t the simplified physics-based formulation of the NMPC algorithm, see Fig. 4.

During the analysis, it was observed that the NNMPC calibration of this use case, being based on a comprehensively tuned FF-ANN, tends to be more or equally robust w.r.t. the benchmarking tachometric controller and NMPC, in case of reasonable variations of the

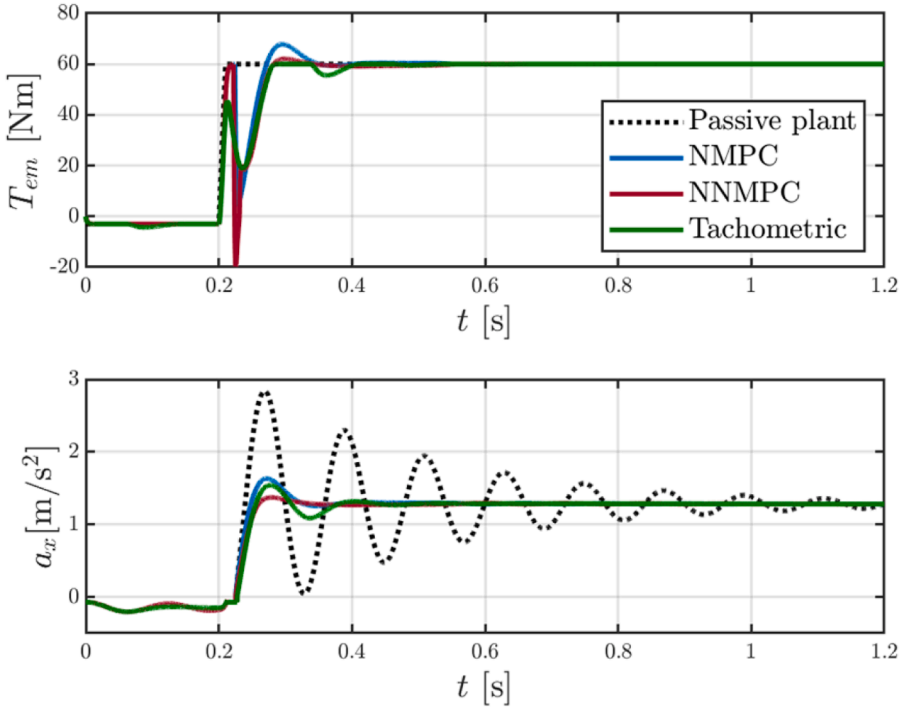


Fig. 11. Time domain results for use case 1, for the nominal EV parametrisation along the considered tip-in test: comparison of the passive EV with the same vehicle with the proposed NNMPC strategy, the benchmarking tachometric controller and NMPC anti-jerk algorithm.

Table 4
Use case 1: comparison of the relevant KPIs, where the bold fonts are used to indicate the configuration providing the best overall performance, measured by J_{opt}.

Controller	$VDV_{a_x^*}$ [m/s$^{1.75}$]	$RMS_{a_x^*}$ [m/s^2]	Δ_{v_x} [km/h]	$\Delta_{t,a}$ [s]	$IACA$ [Nm]	J_{opt} [-]
Without torque ripple and sensor signal noise						
Passive vehicle	0.536	0.397	-	0.035	-	66.15
Tachometric	0.307	0.163	0.043	0.044	4.260	38.39
NMPC	0.322	0.167	0.021	0.038	2.851	37.33
NNMPC	0.308	0.152	0.042	0.038	3.238	**35.97**
With torque ripple and sensor signal noise						
Passive vehicle	0.523	0.389	-	0.035	-	67.59
Tachometric	0.302	0.159	0.041	0.043	4.595	37.98
NMPC	0.301	0.159	0.025	0.041	3.528	36.73
NNMPC	0.300	0.148	0.042	0.042	3.772	**35.88**

plant parameters, which confirms the potential of the proposed NNMPC algorithm. In this respect, dedicated simulations were carried out by including the effect of a motor torque ripple of significant magnitude, according to the formulation in [15], as well as high and purposely unfiltered signal noise on the wheel speed measurements, see the results in Fig. 12 and the KPIs in the bottom section of Table 4.

5.3. Use case 2: adaptation to different operating conditions

Fig. 13 presents the offline analysis conducted on the considered 120 Nm tip-in test, showing the potential effects of the NNMPC adaptation process. Subplots (a) report the a_x profiles for each adaptation step, while the spider chart in Fig. 13(b) includes the respective KPIs. Although being substantially better than the passive configuration, which confirms the robustness of the baseline version of the controller, the EV with the nominal NNMPC is characterized by an evident overshoot following the swift torque increase. Such peak is significantly attenuated by the configuration with the updated neural network, and is absent once also the cost function weights are updated. Such observations are confirmed by the KPIs in Fig. 13(b), in which the fully updated configuration provides a remarkable improvement in J_{opt}, $VDV_{a_x^*}$, and $RMS_{a_x^*}$, without losing in terms of longitudinal acceleration performance, see the absence of degradation of Δv_x and Δ_{t,a_x}.

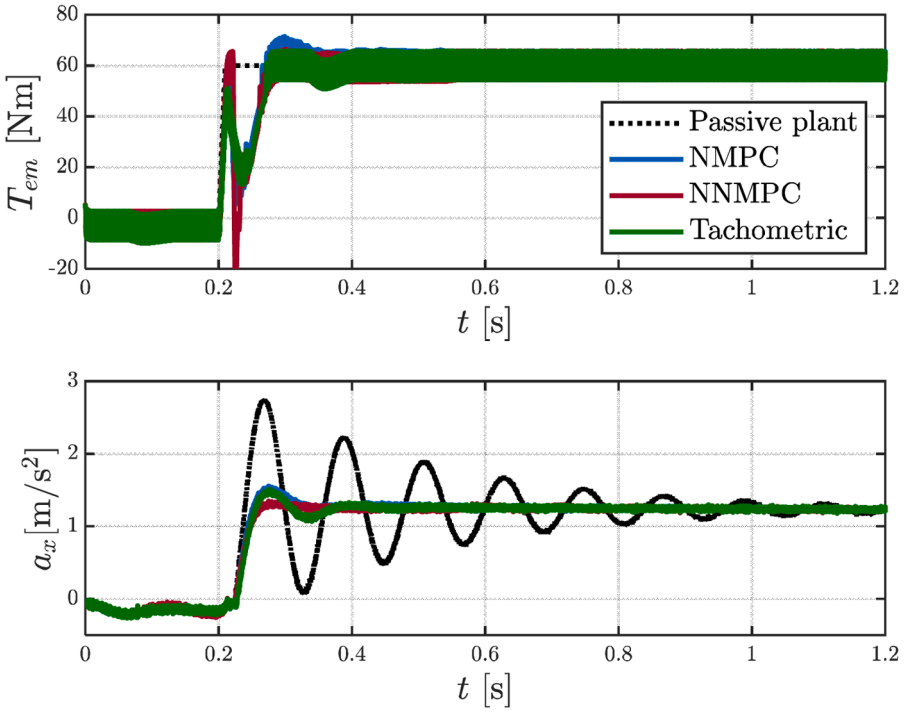


Fig. 12. Time domain results for use case 1 in presence of torque ripple and wheel speed sensor noise, for the nominal EV parametrisation along the considered tip-in test: comparison of the passive EV with the same vehicle with the proposed NNMPC strategy, the benchmarking tachometric controller, and the benchmarking NMPC anti-jerk algorithm.

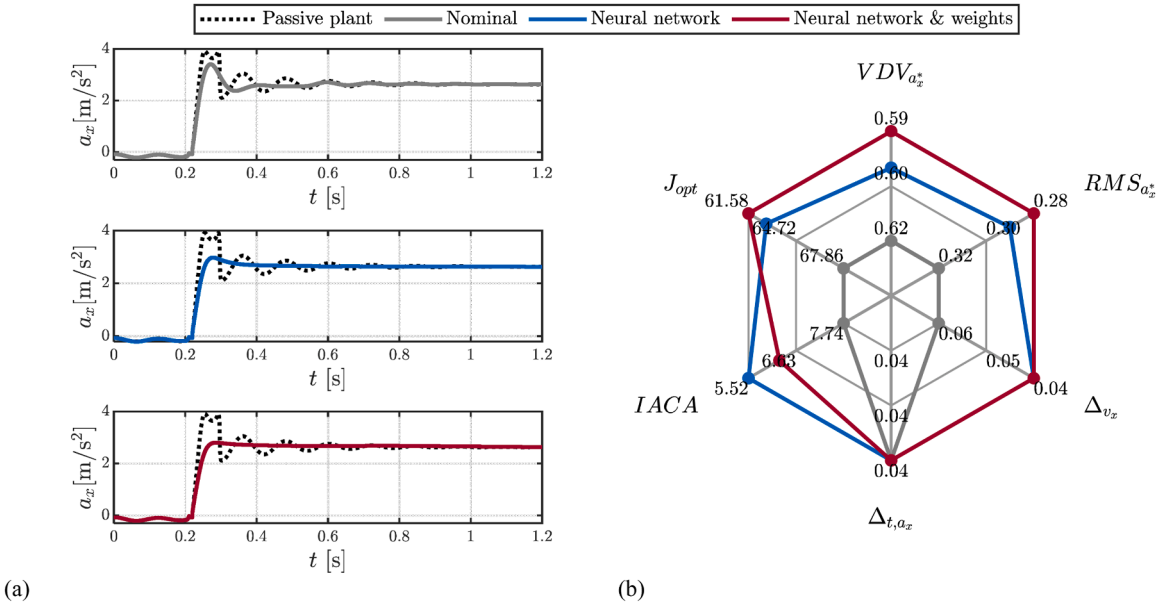


Fig. 13. Offline controller update of Use case 2. Comparison of the: (a) time domain results; and (b) KPIs, for the passive EV, and the same EV with: the NNMPC algorithm with the initial prediction model training ('Nominal'); the NNMPC algorithm with the offline updated FF-ANN and nominal cost function weights ('Neural network'); and the NNMPC with the offline updated FF-ANN and cost function weights ('Neural network & weights').

Fig. 14(a), referring to the online NNMPC update, reports the EV speed profile during the considered sequence of tip-ins / tip-outs, with indication of the time values at which: i) the FF-ANN re-training is initiated by the 'Condition check' block in Fig. 1 and Fig. 10 ('FF-ANN online training (start)' in the plot); ii) the new network calibration is available ('FF-ANN online training (end)'); and iii) the weight optimization is completed ('Online weights optimization (end)'). For the selected tip-in test, Fig. 14(b) reports the KPI percentage difference, Δ_{ind}, associated with the online adaptation w.r.t the offline case, which is defined as:

$$\Delta_{ind} = \frac{ind_{off} - ind_{on}}{ind_{off}} 100 \tag{31}$$

where the notation *ind*, with $ind = VDV_{a_x^*}$, $RMS_{a_x^*}$, Δ_{t,a_x}, $IACA$, and J_{opt}, refers to the considered indicator; and the subscripts 'on' and 'off' refer to the KPI values after the online and offline adaptations. Interestingly, although the online updated FF-ANN presents a good match in terms of predicted accelerations w.r.t. the simulated plant, the NNMPC with the sole online adaptation of the network experiences ~5% and ~20% increments of $VDV_{a_x^*}$ and $RMS_{a_x^*}$ w.r.t. the corresponding offline case. Actually, the resulting controller with the initial weights and online updated network provides worse performance than the nominal NNMPC in Fig. 11. However, once also the cost function weights are updated, the online adaptation provides excellent KPIs, which are summarized by the positive ~5% $\Delta_{J_{opt}}$ value, i.e., an improvement w.r.t. to the offline configuration. The different behaviour of the online and offline update cases is caused by the accelerated learning parametrisation of the online set-up, as well as the inevitable variability of the operating conditions of the EV, e.g., in terms of vehicle speed, during the online update process. Hence, the important preliminary conclusions from this use case are that: i) the online NNMPC update process is feasible and impactful; ii) to be effective, the online update must involve both the network and cost function weights.

5.4. Use case 3: adaptation to varying vehicle parameters

To identify the sensitivity of the vehicle response to the variation of the individual drivetrain and EV parameters, for the tip-in test of use case 1, Fig. 15 plots the a_x profile for the passive EV, and the same vehicle controlled by the non-adaptive NNMPC with the network trained along the UDDS. For example, in the increased backlash case of Fig. 15(b), the initial tip-in response of both the passive and active cases exhibits a marginal delay w.r.t the nominal plant. The delay occurs because the larger backlash requires increased rotation of the transmission shaft before the mechanical contact and torque transmission occur, leading to a higher relative angular velocity at the gear tooth contact instant, and a higher initial a_x peak.

Nonetheless, the active vehicle response with modified backlash remains rather aligned to that of the nominal plant, which is also the case for the modified tyre parametrization in Fig. 15(c), where the passive EV exhibits a lower initial a_x peak and higher damping. The most significant controller performance degradation is observed for increased τ_{em}, see Fig. 15(d), in which – unlike the other cases – the nominal NNMPC fails to attenuate the initial a_x oscillations. Therefore, the time profiles in Fig. 16 focus on the increased motor time constant scenario, while the KPI comparison, reported in Fig. 17, is shown for the individual variation of the three parameters,

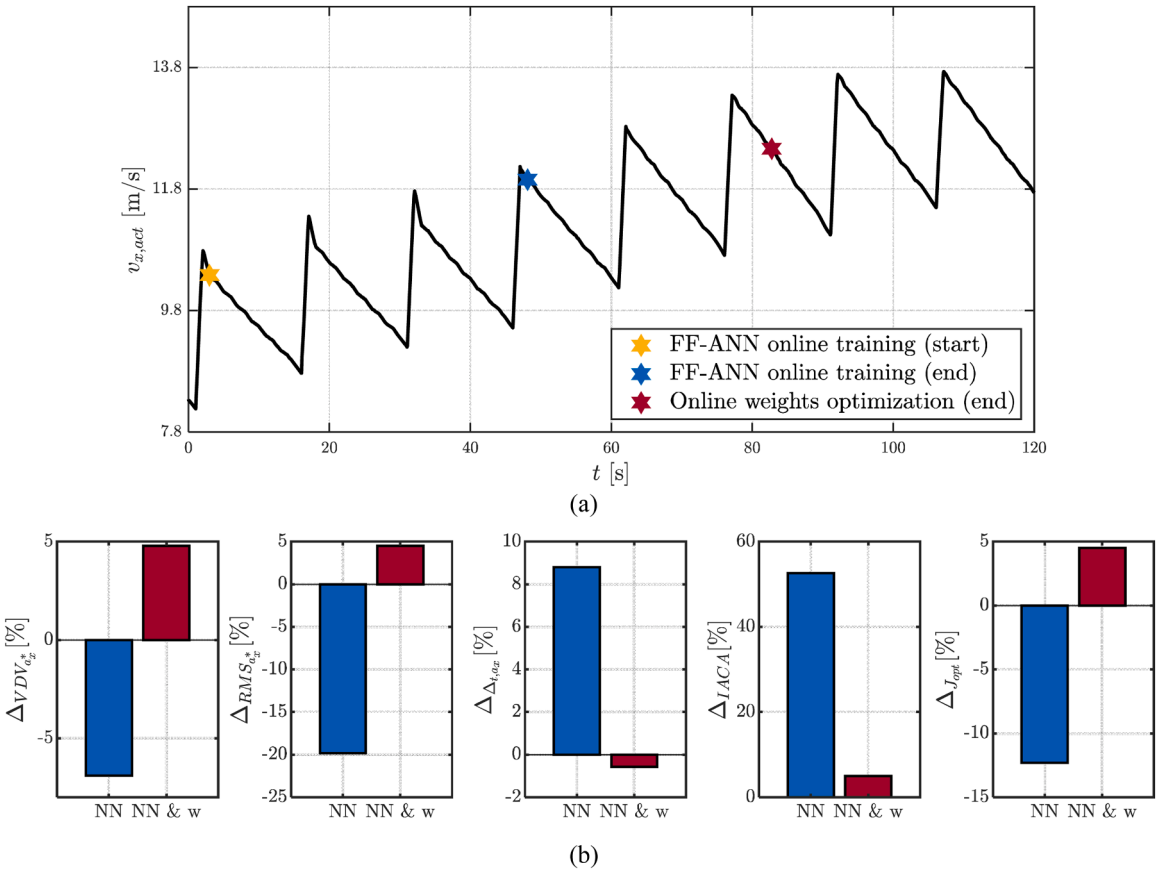


Fig. 14. (a) Longitudinal speed profile along the online adaptation process; and (b) KPI percentage difference between the offline and online adaptations. 'NN': update of the sole neural network prediction model; 'NN & w': update of the neural network prediction model and optimization of the NNMPC cost function weights.

with the KPIs being computed across the entire driving cycle.

In the example in Fig. 16, the online verification and training process is based on an $RMSE_y$ condition check that is carried out every 60 s, by using data collected along the same time interval. Therefore, in Fig. 16(b), at $t = 60$ s, because of the internal model mismatch, $RMSE_y$ exceeds $RMSE_{th}$, and the FF-ANN retraining starts by using the first 60 s of data samples. The network re-training is completed at ~115 s, see the flag variable Tr_{state} in Fig. 16(c), while the subsequent weight optimization process ends at ~140 s, see the Opt_{state} commutation in the same subplot. Hence, in the check at 180 s in Fig. 16(b), $RMSE_y$, which is not re-calculated during the update process and thus is kept constant in the figure in the re-training and optimization phases, becomes lower than $RMSE_{th}$, which confirms the successful FF-ANN adaptation. As shown by the a_x profiles in Fig. 16(d) and the KPI comparison in Fig. 17, also in this use case the optimization of the NNMPC cost function weights is essential for exploiting the benefit of the adaptation process.

5.5. *Use case 4: controller auto-tuning for different vehicle application and operating conditions*

Fig. 18 illustrates the pulsating torque profile of the considered sequence of tip-ins / tip-outs along which the proof-of-concept online auto-tuning of the controller for the new vehicle application is preliminarily demonstrated. In this example, the $RMSE_y$ value is computed every 3 s, and, since it already exceeds $RMSE_{th}$ following the first tip-in and tip-out application, the FF-ANN update is initiated at 3 s, by using the new training dataset collected up to that point, see Fig. 18(a)-(b). The online training process is then completed at ~40 s, see the commutation of Tr_{state} in Fig. 18(c), while the weight optimization finishes at ~80 s, as confirmed by the Opt_{state} profile. As, following the re-training process, $RMSE_y$ remains below the threshold for the remainder of the simulation, indicating the successful adaptation to the new vehicle configuration, no further online training is requested to the parallel pool. The marginal variation in the loss function towards the end of the simulation can be attributed to the different vehicle conditions, e.g., in terms of vehicle speed, w.r.t. those of the online training. The longitudinal acceleration profiles in Fig. 18(d), referring to the three stages of the adaptation process, confirm that major benefits are achievable through the double adaptation routine.

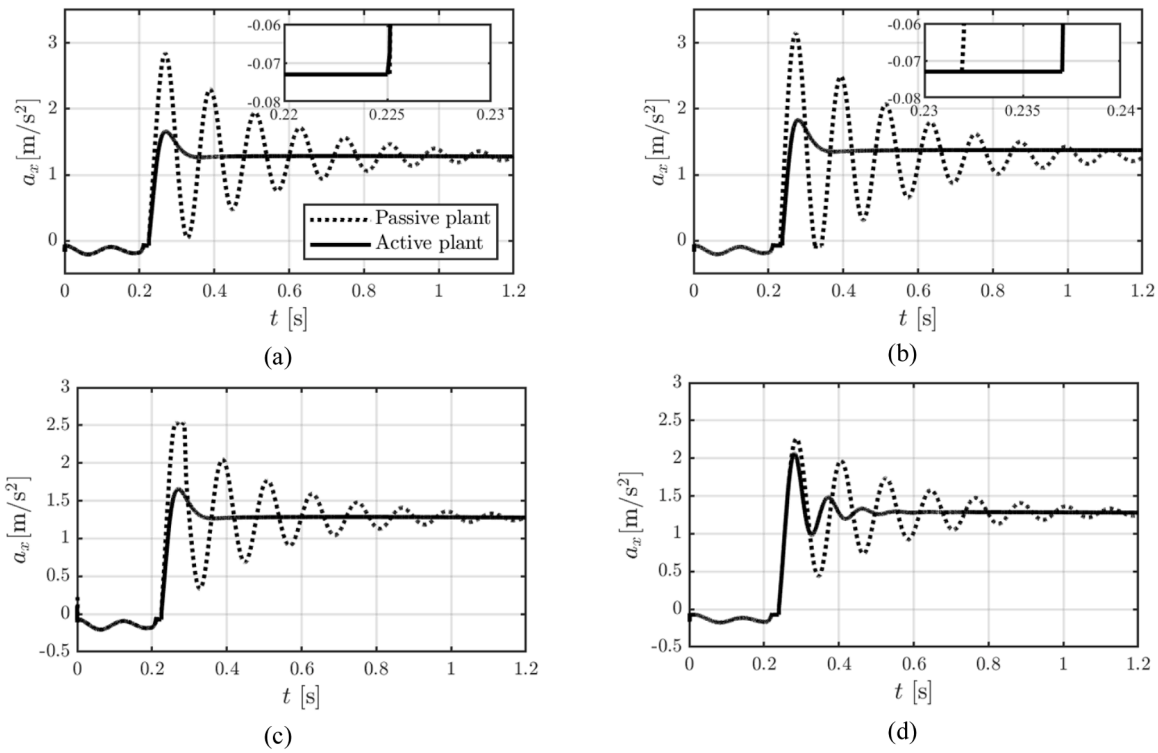


Fig 15. Time domain results along the tip-in test of use case 1, for the passive EV ('Passive plant'), and the controlled EV with the non-adaptive NNMPC ('Active plant') for: (a) Nominal plant model; (b) Increased drivetrain backlash; (c) Different tyre parameters; and (d) Increased motor time constant.

5.6. *Robust stability verification*

An important feature of any controller is the capability of providing robust stability, which is usually referred to as the stability in presence of plant parameter variations. In case of a self-adaptive implementation like the one of this research, such property is even more relevant and complex to verify, since not only the plant, but also the controller itself, are characterised by parametric variations. In fact, in the specific architecture, both the NN parameters (i.e., the weights and biases) and controller cost function weights are subject to automated adaptations, when significant mismatches are detected by the online algorithm in Fig. 10. Moreover, to increase the challenge, for NMPC and NNMPC implementations, there is not an available theoretical methodology to formally verify control system stability in the design phase, e.g., see the stability discussion in [67].

As a consequence, in this proof-of-concept analysis, control system stability is verified through Monte Carlo simulations. Each Monte Carlo analysis includes 500 test scenarios, defined by a combination of parameter values randomly drawn from the respective distributions in Fig. 19, covering α, τ_{em}, C_{k_x} (the reported values refer to the nominal vertical tyre load F_{nom}), and the initial vehicle speed $v_{x,0}$ of the selected manoeuvre, i.e., a tip-in test with a final 60 Nm motor torque request. Two Monte Carlo analyses were implemented, for: i) the NNMPC calibration optimised for the nominal plant; and ii) an updated NNMPC calibration, resulting from the controller adaptation – in terms of its NN prediction model and cost function weights – to radically varied plant parameters, according to the Fig. 10 architecture. For example, for obtaining the controller setting in ii), the vehicle mass is increased by over 500 kg; the front and rear semi-wheelbases are varied by >10 cm, and the centre of gravity height is varied by ~30%, together with the aerodynamic drag coefficient and the vehicle frontal area; the drivetrain backlash size is doubled, and the torsional stiffness is increased by 10%; the torque time constant of the electric powertrain is increased by an order of magnitude, while the mass moment of inertia of the motor rotor is incremented by a factor 3, concurrently with a ~6% variation of the wheel mass moment of inertia; and the longitudinal tyre force characteristics are varied from those of Tyre 1 to those of Tyre 2 in Fig. 3. Within each of the two Monte Carlo analyses, neither the NN prediction model nor the controller settings are varied, while the plant model parametrisation is changed according to the Monte Carlo parameter distribution. The results were also generated for the Passive case, i.e., without anti-jerk controller, with the same parameter distributions.

The Monte Carlo results show that in both cases – with the nominal and updated controllers – the control system provides reliably stable behaviour, with swift drivetrain oscillation damping, and average $VDV_{a_x^*}$ and $RMS_{a_x^*}$ values that are at least halved w.r.t. those of the corresponding passive EV configuration, and with a significantly reduced spread of the KPIs across the individual simulations, which is a sign of intrinsic control system robustness. The stability properties are also confirmed by the time profiles of the controller cost function during the Monte Carlo simulations, showing consistent convergence to zero according to the Lyapunov theory. The

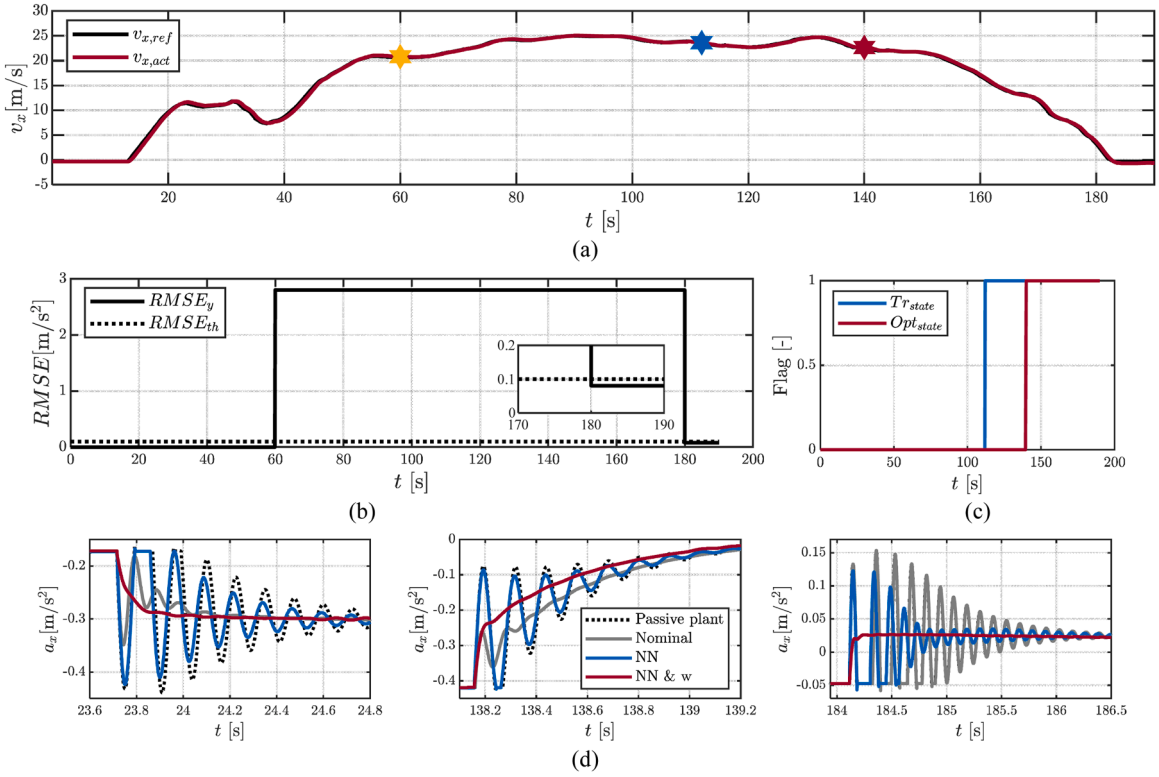


Fig 16. Use case 3 results: (a) Reference ($v_{x,ref}$) and actual ($v_{x,act}$) EV speeds – the latter reported for the 'NN & w' configuration – along the UDDS, with the stars indicating the beginning of the FF-ANN re-training, the end of the network re-training, and the completion of the control weight optimization; (b) $RMSE_y$ and $RMSE_{th}$ profiles; (c) Tr_{state} and Opt_{state} profiles; and (d) a_x profiles for the passive plant, and the same EV including the NNMPCs without adaptations ('Nominal'), with the only FF-ANN adaptation ('NN'), and with both the FF-ANN and cost function weight adaptation ('NN & w'), for three UDDS sections.

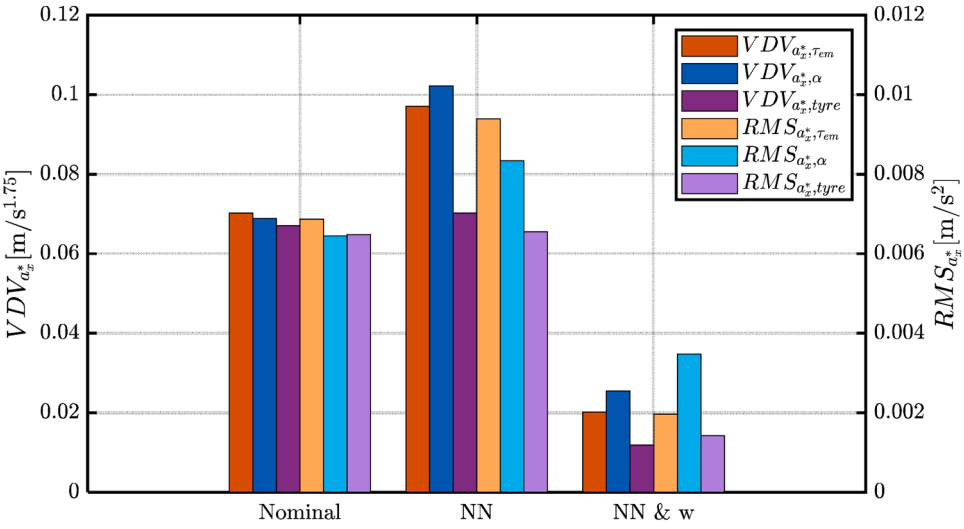


Fig 17. $VDV_{a_x^*}$ and $RMS_{a_x^*}$ comparison for the parameter variations and online updates of use case 3 along the whole UDDS, where the subscript τ_{em} refers to the increased motor torque time constant, α to the increased backlash, and *tyre* to the different tyre characteristics. 'Nominal': NNMPCs without adaptations; 'NN': NNMPCs with the only FF-ANN adaptation; and 'NN & w': NNMPCs with both the FF-ANN and cost function weight adaptations.

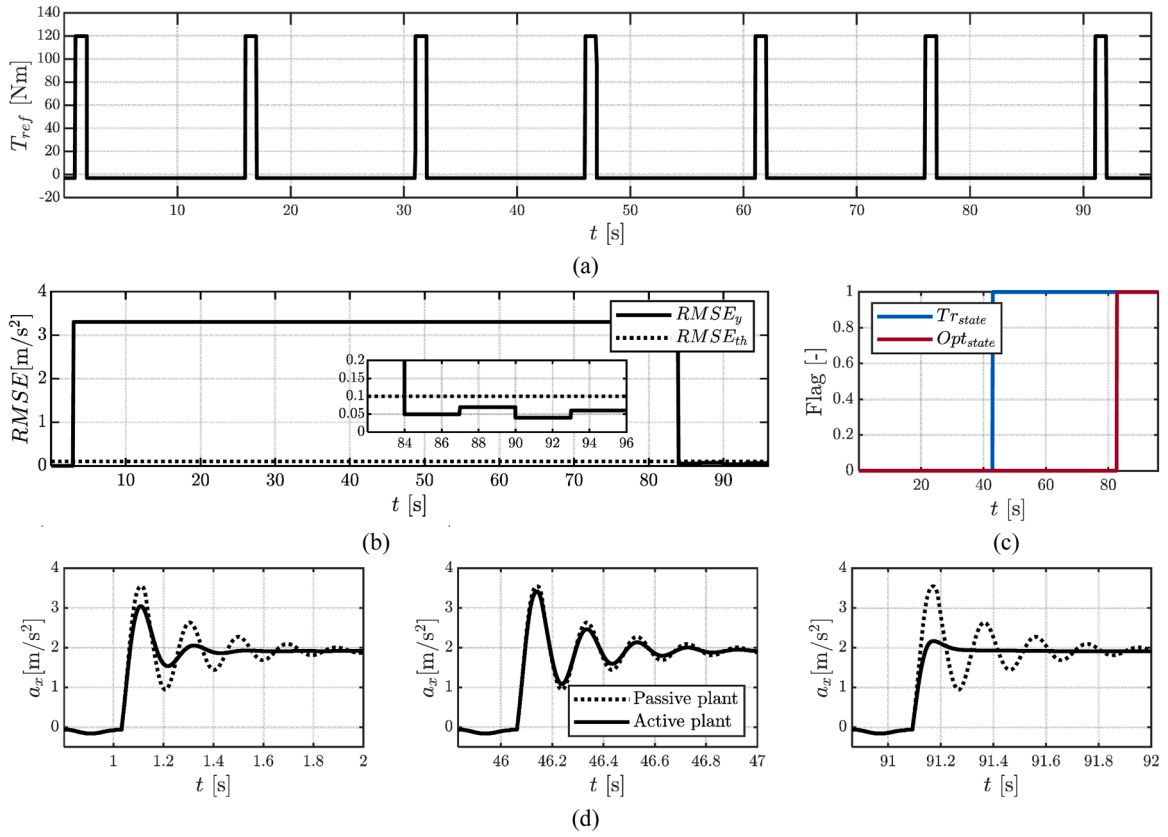


Fig. 18. Use case 4 results: (a) T_{ref} profile along the sequence of tip-ins / tip-outs; (b) $RMSE_y$ and $RMSE_{th}$ profiles; (c) Tr_{state} and Opt_{state} profiles; and (d) Performance comparison of the passive plant, and the same EV including the NNMPCs before the adaptations (left), after the FF-ANN re-training (centre), and after the weight optimization (right).

controller activates itself when the torsional drivetrain oscillations are initiated, and deactivates when these are appropriately damped. In conclusion, the Monte Carlo results confirm the robust stability of the self-adaptive controller, also in case of extreme adaptations. Future developments will target the integration of the verification of the stability aspects in the online adaptation architecture in Fig. 10.

5.7. *Sensitivity to control parameters and proof-of-concept real-time implementability*

Table 5 reports the sensitivity results to the size of the FF-ANN prediction model used by the NNMPC, i.e., the number of hidden layers and neurons are varied to identify the trade-off between resulting control system performance and complexity / computational load. The selected manoeuvres are the tip-in and tip-out to/from 60 Nm, with an initial vehicle speed of 30 km/h. The following FF-ANN architectures are considered: i) a configuration with two hidden layers with 16 neurons each, referred to as '16-16' in the table; ii) a configuration with two hidden layers with 32 neurons each, indicated as '32-32'; and iii) a configuration, '32-64-32', with three hidden layers with 32, 64, and 32 neurons. The offline learning set-up is the same for all network architectures. Each controller was individually calibrated with the cost function weight optimization routine in Section 4, applied to the tip-in test. For both manoeuvres and all controllers, including the benchmarking NMPC, the results are shown for: a) the simulated version of the algorithms (the so-called 'controller simulation set-up'), with $T_s = 1$ ms; and b) the respective proof-of-concept real-time implementation ('controller real-time set-up') on a dSPACE MicroAutoBox III unit (1.4 GHz, 64 Mb flash memory, see Fig. 20). For the control configurations in b), T_s was selected to be the minimum possible to provide real-time capability also in the worst-case scenario (i.e., the one requiring the longest time to output the control action) in terms of turn-around time, which is the time required by the specific rapid control prototyping unit to generate the control input.

The following conclusions can be derived:

- In the simulation set-up, the NNMPC implementations provide consistently better performance than the benchmarking NMPC operating at the same T_s (1 ms), with J_{opt} reductions exceeding 6.5% during the tip-in, and 7.5% during the tip-out.
- For a fixed and sufficiently low T_s, increasing the FF-ANN size tends to improve the results up to some extent, as confirmed by the 16.5% and 6.4% J_{opt} reductions for the simulation set-up of '32-32' w.r.t. the one of '16-16', during the selected tip-in and tip-out.

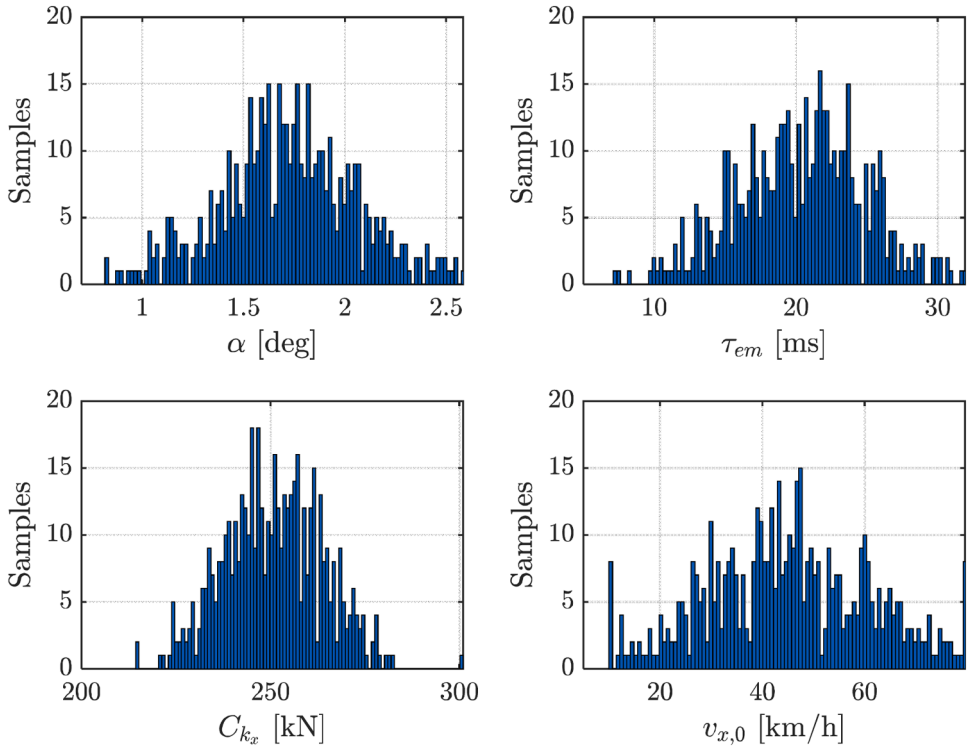


Fig. 19. Parameter distributions of the implemented Monte Carlo analyses.

Table 5
Sensitivity analysis to the variation of the FF-ANN size, along the nominal tip-in and tip-out tests from 30 km/h, where – within each section of the table – the bold fonts are used to indicate the configuration providing the best overall performance, measured by J_{opt}.

Controller	FF-ANN set-up	T_s [ms]	$VDV_{a_x^*}$ [m/s$^{1.75}$]	$RMS_{a_x^*}$ [m/s^2]	Δv_x [km/h]	Δ_{t,a_x} [s]	$IACA$ [Nm]	J_{opt} [-]	$[W_{\Delta\dot{\theta}}, W_{T_{em}}, W_{T_{corr}}]$
Tip-in test; controller simulation set-up									
Passive plant	-	1	0.523	0.389	0	0.035	0	65.23	-
NMPC	-	1	0.313	0.162	-5.3e-5	0.048	3.866	32.49	[1004.4, 0.4, 35]
NNMPC	16-16	1	0.292	0.151	0.041	0.052	2.440	30.41	[1075.8, 86.1, 130.3]
NNMPC	32-32	1	0.264	0.116	0.072	0.089	5.254	**26.10**	[564.3, 33.9, 0.001]
NNMPC	32-64-32	1	0.261	0.120	0.084	0.078	4.130	26.22	[1034.2, 25, 71.1]
Tip-in test; controller real-time set-up									
NMPC	-	2	0.384	0.237	-0.009	0.039	2.121	43.237	[632.2, 0.001, 205.5]
NNMPC	16-16	6	0.282	0.141	0.052	0.059	2.817	**29.069**	[919, 80.8, 27.8]
NNMPC	32-32	18	0.435	0.276	0.014	0.035	3.192	49.943	[1075.8, 24.8, 96.5]
NNMPC	32-64-32	66	0.551	0.427	-0.0014	0.036	1.063	70.509	[527.3, 61.8, 111.7]
Tip-out test; controller simulation set-up									
Passive plant	-	1	0.526	0.349	0	0.027	0	61.404	-
NMPC	-	1	0.310	0.145	0.029	0.05	3.669	30.802	[1004.4, 0.4, 35]
NNMPC	16-16	1	0.289	0.136	0.009	0.054	2.350	28.658	[1075.8, 86.1, 130.3]
NNMPC	32-32	1	0.266	0.111	-0.243	0.485	13.015	26.934	[564.3, 33.9, 0.001]
NNMPC	32-64-32	1	0.271	0.121	0.014	0.08	2.858	**26.381**	[1034.2, 25, 71.1]
Tip-out test; controller real-time set-up									
NMPC	-	2	0.372	0.196	0.012	0.036	2.206	38.668	[632.2, 0.001, 205.5]
NNMPC	16-16	6	0.310	0.144	0.078	0.047	4.888	**30.986**	[919, 80.8, 27.8]
NNMPC	32-32	18	0.293	0.225	-0.003	0.036	8.842	38.214	[1075.8, 24.8, 96.5]
NNMPC	32-64-32	66	0.859	0.606	0.028	0.03	0.686	103.970	[527.3, 61.8, 111.7]

However, for $T_s = 1$ ms, the '32-64-32' configuration provides slightly worse performance during the tip-in test, w.r.t. the '32-32' case, while it manages to marginally improve the tip-out results, with a 2.1% J_{opt} reduction.

- In comparison with the simulation set-up, the real-time controller implementation requires an increase of T_s to ensure that this always exceeds the turn-around time. Since on the specific plant with very fast powertrain torque dynamics the performance is subject to an evident degradation with increasing T_s for all configurations, the selection of the complexity level of the FF-ANN architecture becomes more critical than in the simulation case. In fact, on the specific dSPACE unit, the NMPC anti-jerk

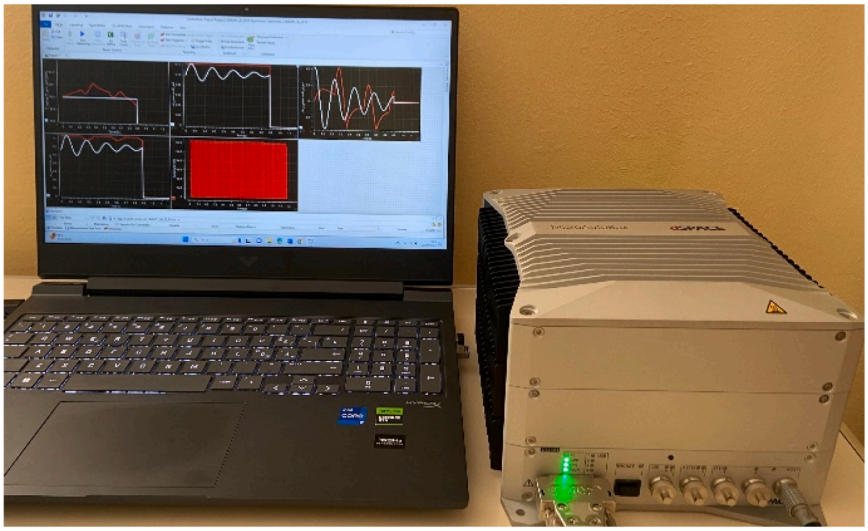


Fig. 20. Real-time implementation and operation of the proposed controllers on a dSPACE MicroAutoBox III unit.

algorithm can run at 2 ms, while the NNMPCs corresponding to '16-16', '32-32', and '32-64-32' respectively need time steps of 6, 18, and 66 ms. Hence, the NNMPC with the network structure with 2 hidden layers of 16 neurons each provides the best performance, with 11.8% and 48.7% J_{opt} reductions in the tip-in w.r.t. to the simulation and real-time set-ups of the benchmarking NMPC, and 0.6% and 24.8% in the tip-out. This is the ultimate confirmation of the NNMPC potential for advanced driveline control.

In terms of the practical implementation of the online self-adaptations, in use cases 2-4, the update was carried out in a parallel pool w.r.t. the one running the main simulation, to highlight that the update process could be swiftly carried out on the same multi-core hardware as the one computing the real-time control inputs. In this respect, the idea could be to carry out a main comprehensive offline network training, like the one of use case 1, and then frequent online network mini-re-trainings, such as those of use cases 2-4, to keep the system up-to-date. Alternatively, more realistically for a short-term real EV implementation, the NNMPC algorithm and the FF-ANN adaptation / cost function weight optimization can be implemented on separate control units: i) a system dedicated to the real-time solution of the NNMPC OCP, and represented by the dSPACE unit in the proof-of-concept implementation of this section; and ii) an in-vehicle personal computer (PC), responsible for the periodic learning update of the relevant neural networks, and control weight optimization, independent from the online operation of the control algorithms. W.r.t. ii), in the preliminary assessment of this study, it was verified that by using the discussed Matlab/Simulink toolchain on a computer with 32 GB RAM, a 2.2 GHz Intel Core i9 processor, and a 1 TB solid-state drive, the tip-in based FF-ANN re-training and weight optimization of use case 2 take respectively 188 s and 157 s, with the online parametrization of the learning process in Table 3. The same network re-training takes 1054 s if carried out with the offline learning parametrization in Table 2, while a comprehensive re-training based on the very broad data set of use case 1 takes 11272 s, e.g., the controller could be updated while the vehicle is at rest. Such durations are all compatible with the proposed functionality, aiming to perform periodic updates, rather than continuous dynamic variations of the NNMPC set-up. The analysis of the practical vehicle implementation of the individualized anti-jerk control updates on a real vehicle platform will be a future development topic.

6. Conclusions

This proof-of-concept study introduced a novel self-adaptive neural network model predictive control (SA-NNMPC) algorithm for the anti-jerk control of electric vehicles with on-board electric powertrains. During vehicle operation, the strategy integrates a real-time update mechanism of both the neural network prediction model and cost function weights.

The preliminary simulations along tip-in / tip-out manoeuvres and driving cycles show that in nominal conditions, in absence of any adaptation, the neural network model more accurately replicates the nonlinear plant behaviour than the typical physics-based drivetrain dynamics formulation of the considered benchmarking NMPC implementation, leading to better controller performance, as confirmed by the lower values of the relevant key performance indicators. Also in absence of any adaptation, the NNMPC algorithm is able to manage reasonable parameter variations of the vehicle plant, still providing a benefit w.r.t. the NMPC configuration as well as a classical tachometric anti-jerk control implementation. The online self-adaptive architecture is effective in handling changes of different nature, as confirmed by the considered use cases, dealing with the variation of: i) the operating conditions w.r.t. those of the initial network training; ii) the main drivetrain and vehicle parameters; and iii) the whole vehicle altogether. An appropriately parameterised version of the NNMPC algorithm could run in real-time on the considered rapid control prototyping unit, still providing clear benefits w.r.t. both the simulation and real-time set-ups of the benchmarking NMPC strategy.

Future developments will focus on: i) the experimental implementation and assessment of the proposed NNMPC architecture,

including its self-adaptation mechanism, on an electric vehicle demonstrator, along a wider range of conditions; and ii) the further development of the control structure, by using physics-informed neural network approaches in the prediction model, and including formal methodologies and online routines to ensure the robust stability of the adapted controller.

CRediT authorship contribution statement

Gianluca Frison: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Fabio Alberti:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Luca Ciravegna:** Writing – review & editing, Validation, Software, Investigation, Data curation, Conceptualization. **Luca Dimauro:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Project administration, Methodology, Investigation, Data curation, Conceptualization. **Aldo Sorniotti:** Writing – review & editing, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the Horizon 2020 Programme of the European Commission under Grant agreement numbers 101138266 (EFFEREST project) and 101096062 (CLIMAFlux project). The authors would like to thank Dr Alessandro Scamarcio for providing the simulation model of the plant.

Data availability

Data will be made available on request.

References

[1] S. De Pinto, P. Camocardi, C. Chatzikomis, A. Sorniotti, F. Bottiglione, G. Mantriota, P. Perlo, On the comparison of 2- and 4-wheel-drive electric vehicle layouts with central motors and single- and 2-speed transmission systems, Energies 13 (2020).
[2] F. Bottiglione, A. Sorniotti, L. Shead, The effect of half-shaft torsion dynamics on the performance of a traction control system for electric vehicles, Proc. Inst. Mech. Eng. Part D J. Autom. Eng. 226 (9) (2012) 1145–1159.
[3] A. Lagerberg, B. Egardt, Backlash estimation with application to automotive powertrains, IEEE Trans. Control Syst. Technol. 15 (3) (2007) 483–493.
[4] D. Hodgson, B.C. Mecrow, S.M. Gadoue, H.J. Slater, P.G. Barrass, D. Giaouris, Effect of vehicle mass changes on the accuracy of Kalman filter estimation of electric vehicle speed, IET Electr. Syst. Transp. 3 (2013) 67–78.
[5] M.J. Griffin, Discomfort from feeling vehicle vibration, Vehicle Syst. Dyn. 45 (7-8) (2007) 679–698.
[6] C.A.M. Makosi, S. Rinderknecht, R. Binz, F. Uphaus, F. Kirschbaum, Implementation of an open-loop controller to design the longitudinal vehicle dynamics in passenger cars, SAE Tech. Pap. (2017), 2017-01-1107.
[7] G. Genta, L. Morello, The Automotive Chassis, 2, Springer, 2009. System Design.
[8] J. Fredriksson, H. Weiefors, B. Egardt, Powertrain control for active damping of driveline oscillations, Vehicle Syst. Dyn. 37 (5) (2002) 359–376.
[9] K. Koprubasi, E.R. Westervelt, G. Rizzoni, E. Galvagno, M. Velardocchia, Experimental validation of a model for control of drivability in a hybrid-electric vehicle, in: ASME International Mechanical Engineering Congress and Exposition 16, 2007, pp. 105–114.
[10] A. Walter, U. Kiencke, S. Jones, T. Winkler, Anti-jerk & idle speed control with integrated sub-harmonic vibration compensation for vehicles with dual mass flywheels, SAE Int. J. Fuels Lubr. 1 (1) (2009) 1267–1276.
[11] P.D. Walker, N. Zhang, Active damping of transient vibration in dual clutch transmission equipped powertrains: a comparison of conventional and hybrid electric vehicles, Mech. Mach. Theory. 77 (7) (2014) 1–12.
[12] H. Fukudome, "Reduction of longitudinal vehicle vibration using in-wheel motors," *SAE Technical Paper*, 2016-01-1668, 2016.
[13] D. Lefebvre, P. Chevrel, S. Richard, An H-infinity-based control design methodology dedicated to the active control of vehicle longitudinal oscillations, IEEE Trans. Control Syst. Technol. 11 (6) (2003) 948–956.
[14] A. Scamarcio, P. Gruber, S. De Pinto, A. Sorniotti, Anti-jerk controllers for automotive applications: a review, Annu Rev. Control 50 (2020) 174–189.
[15] A. Scamarcio, C. Caponio, M. Mihalkov, P. Georgiev, J. Ahmadi, K.M. So, D. Tavernini, A. Sorniotti, Predictive anti-jerk and traction control for V2X connected electric vehicles with central motor and open differential, IEEE Trans. Veh. Technol. 72 (6) (2023) 7221–7239.
[16] A. Koch, J. Brauer, J. Falkenstein, Drivability optimization of electric vehicle drivetrains for brake blending maneuvers, World Electric Veh. J. 13 (11) (2022).
[17] V. Vidal, P. Stano, G. Tavolo, M. Dhaens, D. Tavernini, P. Gruber, A. Sorniotti, On Pre-Emptive In-Wheel Motor Control for Reducing the Longitudinal Acceleration Oscillations Caused by Road Irregularities, IEEE Trans. Veh. Technol. 71 (9) (2022) 9322–9337.
[18] P. Stano, D. Lazzarini, S. Santoro, M. Mihalkov, U. Montanaro, A. Vigliani, A. Ferrara, M. Dhaens, A. Sorniotti, On-board electric powertrain control for the compensation of the longitudinal acceleration oscillations caused by road irregularities, Mech. Mach. Theory. 202 (2024) 105759.
[19] I. Youn, E. Ahmad, Anti-jerk optimal preview control strategy to enhance performance of active and semi-active suspension systems, Electronics 11 (10) (1657) 2022.
[20] E. Ahmad, I. Youn, Performance Improvement during Attitude Motion of a Vehicle Using Aerodynamic-Surface-Based Anti-Jerk Predictive Controller, Sensors 23 (12) (2023) 5714.
[21] E. Ahmad, I. Youn, Performance improvement of a vehicle equipped with active aerodynamic surfaces using anti-jerk preview control strategy, Sensors 22 (20) (2022) 8057.
[22] X. Zhu, H. Zhang, D. Cao, Z. Fang, Robust control of integrated motor-transmission powertrain system over controller area network for automotive applications, Mech. Syst. Signal. Process. 58 (1) (2015) 15–28.
[23] M. Berriri, P. Chevrel, D. Lefebvre, Active damping of automotive powertrain oscillations by a partial torque compensator, Control Eng. Pract. 16 (2008) 874–883.

[24] K.J. Figel, M. Schultalbers, F. Svaricek, Review and experimental evaluation of models for drivability simulation with focus on tire modeling, Forschung Im Ingenieurwesen/Eng. Res. 83 (2) (2019) 105–118.
[25] N. Amann, J. Böcker, F. Prenner, Active damping of drive train oscillations for an electrically driven vehicle, IEEE/ASME Trans. Mech. 9 (4) (2004) 697–700.
[26] J. Böcker, N. Amann, B. Schulz, Active suppression of torsional oscillations, IFAC Proc. 37 (14) (2004) 319–324.
[27] P. Reddy, K. Darokar, D. Robinette, M. Shahbakhti, M. Ravichandran, J. Doering, Backlash size estimation in automotive drivelines, in: CCTA 2020 - 4th IEEE Conference on Control Technology and Applications, 2020, pp. 201–206.
[28] K. Darokar, P. Reddy, J. Furlich, D. Robinette, M. Shahbakhti, M. Ravichandran, J. Doering, Automotive backlash position estimator for driveline jerk control, in: 4th IEEE Conference on Control Technology and Applications (CCTA), 2020, pp. 88–93.
[29] P. Reddy, M. Shahbakhti, M. Ravichandran, J. Doering, Real-Time Estimation of Backlash Size in Automotive Drivetrains, IEEE/ASME Trans. Mech. 27 (5) (2022) 3362–3372.
[30] P. Reddy, M. Shahbakhti, M. Ravichandran, J. Doering, Real-time predictive clunk control using a reference governor, Control Eng. Pract. 135 (2023) 105489.
[31] J. Wang, H. Xu, X. Hou, C. Du, Q. Zhou, Cross-domain collaborative oscillation control strategy for a hybrid driveline based on the integration of a notch filter, PI filter, and backlash estimator, IFAC-PapersOnLine 54 (2021) 540–545.
[32] B. Houska, H.J. Ferreau, M. Diehl, ACADO toolkit - An open-source framework for automatic control and dynamic optimization, Opt. Control Appl. Methods 32 (2011) 298–312.
[33] A. Scamarcio, M. Metzler, P. Gruber, S. De Pinto, A. Sorniotti, Comparison of anti-jerk controllers for electric vehicles with on-board motors, IEEE Trans. Veh. Technol. 69 (10) (2020) 10681–10699.
[34] A. Norouzi, H. Heidarifar, H. Borhan, M. Shahbakhti, C.R. Koch, Integrating machine learning and model predictive control for automotive applications: a review and future directions, Eng. Appl. Artif. Intell. 120 (2023) 105878.
[35] N.A. Spielberg, M. Brown, J.C. Gerdes, Neural network model predictive motion control applied to automated driving with unknown friction, IEEE Trans. Control Syst. Technol. 30 (5) (2022) 1934–1945.
[36] P. Stano, U. Montanaro, D. Tavernini, M. Tufo, G. Fiengo, L. Novella, A. Sorniotti, Model predictive path tracking control for automated road vehicles: A review, Annu Rev. Control 55 (2023) 194–236.
[37] M. Rokonuzzaman, N. Mohajer, S. Nahavandi, S. Mohamed, Model predictive control with learned vehicle dynamics for autonomous vehicle path tracking, IEEe Access. 9 (2021) 128233–128249.
[38] Z. Gao, W. Wen, Y. Xing, A. Tsourdos, An integrated framework for autonomous driving planning and tracking based on NNMPC considering road surface variations, IEEE Trans. Intell. Veh. (2024) in press.
[39] N.A. Spielberg, M. Brown, N.R. Kapania, J.C. Kegelman, J.C. Gerdes, Neural network vehicle models for high-performance automated driving, Sci. Robot. 4 (28) (2019).
[40] D.C. Gordon, A. Winkler, J. Bedei, P. Schaber, J. Andert, C.R. Koch, Introducing a Deep Neural Network-based Model Predictive Control Framework for Rapid Controller Implementation, in: American Control Conference, 2024.
[41] D.C. Gordon, A. Norouzi, A. Winkler, J. McNally, E. Nuss, D. Abel, M. Shahbakhti, J. Andert, C.R. Koch, End-to-end deep neural network based nonlinear model predictive control: experimental implementation on diesel engine emission control, Energies 15 (2022).
[42] E.A. Antonelo, E. Camponogara, L.O. Seman, J.P. Jordanou, E.R. de Souza, J.F. Hübner, Physics-informed neural nets for control of dynamical systems, Neurocomputing 579 (2024) 127419.
[43] L.D. McClenny, U.M. Braga-Neto, Self-adaptive physics-informed neural networks, J. Comput. Phys. 474 (2023) 111722.
[44] T. Liu, J. Zhao, J. Huang, Z. Li, L. Xu, B. Zhao, Research on model predictive control of autonomous underwater vehicle based on physics informed neural network modeling, Ocean Eng. 304 (2024) 117844.
[45] Y. Li, L. Liu, Physics-Informed Neural Network-Based Nonlinear Model Predictive Control for Automated Guided Vehicle Trajectory Tracking, World Electr. Vehicle J. 15 (10) (2024) 460.
[46] D. Slama, A. Nonnenmacher, T. Irawan, The Software-Defined Vehicle, O'Reilly Media, Inc., 2023.
[47] G. Costa, J. Pinho, M.A. Botto, P.U. Lima, Online learning of MPC for autonomous racing, Rob. Auton. Syst. 167 (2023) 104469.
[48] B. Ma, X. Guo, P. Li, Adaptive energy management strategy based on a model predictive control with real-time tuning weight for hybrid energy storage system, Energy 283 (1) (2023).
[49] A. Parra, A. Zubizarreta, J. Pérez, An energy efficient intelligent torque vectoring approach based on fuzzy logic controller and neural network tire forces estimator, Neural Comput. Appl. 33 (12) (2021) 9171–9184.
[50] M. Abtahi, M. Rabbani, S. Nazari, An automatic tuning MPC with application to ecological cruise control, IFAC-PapersOnLine 56 (10) (2023) 265–270.
[51] T. Roulet, M. Bach, M.H. Skull, AI-based methodology for the calibration of anti-jerk control on electric drives, ATZ Worldwide 126 (2024) 16–21.
[52] L. De Novellis, A. Sorniotti, P. Gruber, Driving modes for designing the cornering response of fully electric vehicles with multiple motors, Mech. Syst. Signal. Process. 64–65 (2015) 1–15.
[53] B. Lenzo, G. De Filippis, A.M. Dizqah, A. Sorniotti, P. Gruber, S. Fallah, W. De Nijs, Torque distribution strategies for energy-efficient electric vehicles with multiple drivetrains, J. Dyn. Syst. Measur. Control 139 (3) (2017) 121004.
[54] G. De Filippis, B. Lenzo, A. Sorniotti, P. Gruber, W. De Nijs, Energy-efficient torque-vectoring control of electric vehicles with multiple drivetrains, IEEE Trans. Veh. Technol. 67 (6) (2018) 4702–4715.
[55] H.B. Pacejka, Tire and Vehicle Dynamics, 3rd ed., Elsevier, Amsterdam, The Netherlands, 2012.
[56] A. Scamarcio, M. Metzler, P. Gruber, A. Sorniotti, Influence of the prediction model complexity on the performance of model predictive anti-jerk control for on-board electric powertrains, Lecture Notes Mech. Eng. (2020) 1593–1603.
[57] S. Jung, T.Y. Kim, W.S. Yoo, Advanced slip ratio for ensuring numerical stability of low-speed driving simulation. Part I: longitudinal slip ratio, Proc. Inst. Mech. Eng. Part D: J. Autom. Eng. 233 (8) (2019) 2000–2006.
[58] E.G. Giakoumis, Driving and Engine Cycles, Springer, Cham, 2017.
[59] A. Apicella, F. Donnarumma, F. Isgrò, R. Prevete, A survey on modern trainable activation functions, Neural Netw. 138 (2021) 14–32.
[60] L. Xiaohui, C. Hong, Z. Huayu, W. Ping, G. Bingzhao, Design of Model Predictive Controller for Anti-Jerk During Tip-in/out Process of Vehicles, in: 30th Chinese Control Conference, 2011.
[61] A. Lagerberg, B. Egardt, Model Predictive Control of Automotive Powertrains with Backlash, IFAC Proc. 38 (1) (2005) 1–6.
[62] M. Batra, J. McPhee, N.L. Azad, Real-time model predictive control of connected electric vehicles, Veh. Syst. Dyn. 57 (11) (2019) 1720–1743.
[63] T. Zhang, Y. Huang, P. Dai, D. Hao, MPC Controller Design For Low Frequency Vehicle Longitudinal Vibration Suppression, in: 2nd International Conference on Control, Automation and Artificial Intelligence, 2017, pp. 123–129.
[64] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, M. Diehl, Acados—a modular open-source framework for fast embedded optimal control, Math. Program. Comput. 14 (3) (2021) 1–37.
[65] Mechanical vibration and shock – Evaluation of human exposure to wholebody vibration. Part 1: General Requirements, BS ISO 2631-1:1997, 2011.
[66] Human response to vibration – Measuring instrumentation. Part 1: General purpose vibration meters, BS EN ISO 8041-1, 20, 2017.
[67] A. Grancharova, T.A. Johansen, Explicit Nonlinear Model Predictive Control – Theory and Applications, Springer, Heidelberg New York Dordrecht London, 2012.