Model Predictive Control with a Neural Network Model of a Formula Student Prototype

Henrique Manuel Caldeira Pimentel Furtado
henrique.furtado@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

November 2020

Abstract

Autonomous vehicles are becoming increasingly popular amongst the general public and Formula Student competitions are encouraging the development of this technology. The interest in this area has increased over the last years, motivated by the added safety and convenience, as well as cost reduction. The objective of this work is to design a controller with the Model Predictive Control (MPC) methodology for a Formula Student prototype that works on the respective competition tracks. The proposed solution consists in implementing a nonlinear MPC with an Artificial Neural Network (ANN) prediction model.A nonlinear prototype dynamical model was presented and developed, on which control algorithms were implemented and simulated. This dynamical model was validated by introducing in its inputs recorded data from the steering angle and velocity and comparing the simulated outputs with real data from the prototype. The dynamical model was then identified, using an ANN based on adequate signals that capture the entire operation envelope of the system. Reference trajectories were parameterized with respect to the arc length of a series of third order polynomials and the contour error relative to the trajectory was defined. A cost function optimizes the evolution of the states predicted by the ANN model, such that the contour error is minimized while the progression along the track is maximized, during the prediction horizon. The developed MPC controller tracked well the reference trajectories. A different ANN is explored, receiving as inputs the reference trajectory and the states of the system and returning as outputs the respective control actions, thus replacing the MPC and the need of optimizing in real-time.

Keywords: Model Predictive Control, Artificial Neural Networks, Formula Student, Autonomous Driving

1. Introduction

1.1. Motivation

In Formula Student competitions, students are challenged to design, build and test a race car according to a specific set of rules stated by Formula Student Germany (FSG) and Formula Society of Automotive Engineers (FSAE). The Formula Student Team of Técnico Lisboa (FST Lisboa) has been developing cars for these competitions since 2001. In 2019, the team began pursuing two projects simultaneously in its 10^{th} generation of cars. It is building a new electric prototype - FST10e - and at the same time empowering FST09e, the car that competed in the summer of 2019, with autonomous features, which will become FST10d.

The performance of the prototypes participating in the competition is evaluated in a series of four separate dynamic events: Acceleration; Skidpad; Autocross and Endurance/Trackdrive & Efficiency.

Following the growing investment from the automotive industry in autonomous driving [1], the organization of FSG has decided that from 2021 onwards all participating vehicles will have to complete the acceleration event in driverless mode and from 2022 onwards, driverless is also mandatory for the skidpad. reinforcing the incentive for FS teams to develop this technology.

1.2. State of the Art

Some FS teams have successfully implemented Model Predictive Control (MPC) in their cars, like AMZ from Zurich [2] and Oxford Brookes Racing [3]. MPC dates back to the 1970s and it started to emerge industrially in the 1980s [4]. In recent years, with the increasing computing power of microprocessors, its use has spread to many other fields including automotive and aerospace, being used for example in power system balancing models [5] and in power electronics [6].

MPC has the advantage of computing the op-

timal solution by using a prediction model which allows the controller to deal with a replica of the system dynamics, improving the control quality [7]. It also has the advantage of allowing constraints on the inputs, outputs, and states of the system. The prediction model typically involves modelling the physics of the system through mathematical expressions. Most MPC algorithms are based on a linear model of the system. When the goal is to maintain the system at a desired steady state (which happens often in industrial processes), rather than moving rapidly between different operating points, a precisely identified linear model is sufficiently accurate around a certain operating point [8]. If the system is highly nonlinear and large frequent disturbances are present, a nonlinear model is necessary to describe the dynamics [9].

In situations where a nonlinear model is required, the task of obtaining an accurate dynamical model is more difficult. Artificial neural networks (ANN) provide an easier way to model complex systems due to their ability to learn and approximate nonlinear functions. These models can then be used as a prediction model for the MPC.

2. Vehicle Dynamics Model

A car is a complex system which can be divided in several subsystems. Figure 1 shows schematically these subsystems and how they interact with each other. The dynamical model developed in this work is built upon existing models of previous FST Lisboa prototypes [10, 11].

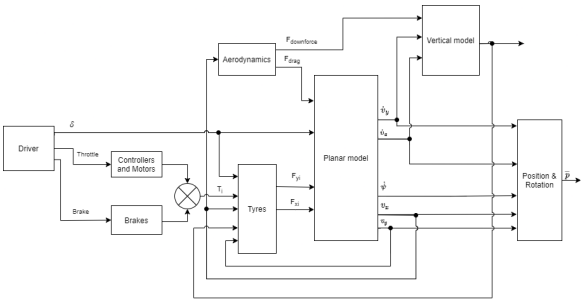

Figure 1: General Structure of the Vehicle Model

There are three subsystems that have a major impact in the forces applied on a car: the powertrain, the aerodynamics and the tyres. The planar model receives these forces and determines the horizontal movement of the vehicle, as well as the accelerations ($\dot{v}_x$ and $\dot{v}_y$). These accelerations generate load transfers, on top of which the aerodynamic loads are added and the vertical load on each tyre (F_z) is then calculated in the vertical model, by simulating the response of the suspension. The tyre model takes the vertical loads on the tyres together with the velocity components and the steering angle (δ) to determine the tyre lateral forces (F_{yi}) given to the planar model. The driver controls the steering angle and the reference velocity, which is then converted to a certain percentage of throttle or brake pedal.

2.1. Planar Model

In Figure 2 the forces applied on the car in the planar model are illustrated.

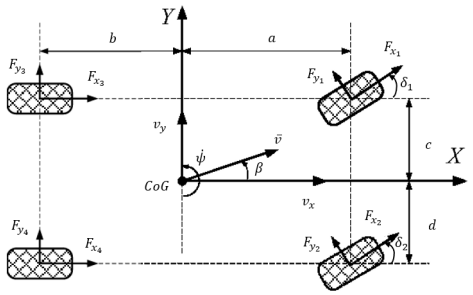

Figure 2: Forces Applied on the Vehicle - Planar Model

Only the front wheels are steerable and each wheel has a different steering angle, due to the Ackermann geometry. The track width is the same at the front and at the rear. Combining the forces in Figure 2 with force and moment equilibrium, the following equations can be obtained:

$$\dot{v}_x = v_y \cdot \dot{\psi} - \frac{1}{m}\left[\sum_{i=1}^{2} {F_{yi}}^F sin(\delta_i) - \sum_{i=1}^{2} {F_{xi}}^F cos(\delta_i) - F_x^R\right] \tag{1a}$$

$$\dot{v}_y = -v_x \cdot \dot{\psi} - \frac{1}{m}\left[\sum_{i=1}^{2} {F_{yi}}^F cos(\delta_i) + F_y^R + \sum_{i=1}^{2} {F_{xi}}^F sin(\delta_i)\right] \tag{1b}$$

$$\ddot{\psi} = \frac{1}{I_\psi} a\left[\sum_{i=1}^{2} {F_{yi}}^F cos(\delta_i) + {F_{xi}}^F sin(\delta_i)\right] - \frac{1}{I_\psi} b F_y^R \tag{1c}$$

where the superscripts F and R denote the total forces in the respective component at the front and at the rear and I_ψ is the inertia around the z-axis.

2.2. Validation

The vehicle model receives as inputs the steering angle and the reference velocity and it outputs the trajectory, lateral acceleration and yaw rate. Data from a skidpad event is used to validate the model, with the steering encoder recording the steering angle and the GPS recording the velocity of the vehicle, which is given as the reference velocity to the model. The skidpad consists of 2 circles to the right followed by to circles to the left, all with the same radius. Negative values of steering angle, lateral acceleration and yaw rate represent the car cornering

to the right while positive values represent the car cornering to the left.

The next step is to give these inputs to the model and compare the simulation outputs with the real data. Figure 3 shows the lateral acceleration simulated and experimental. Figure 4 shows that the simulated model follows closely the yaw rate of the real car.

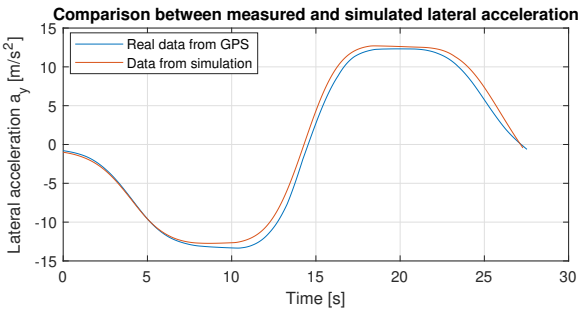

Figure 3: Comparison between measured and simulated lateral acceleration

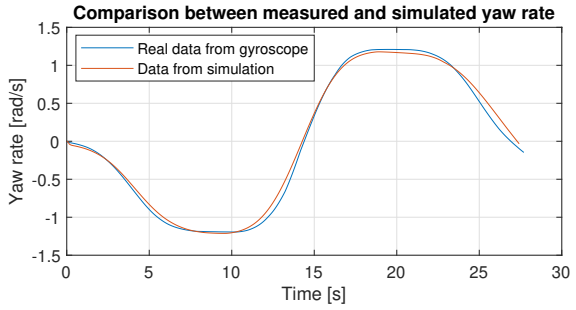

Figure 4: Comparison between measured and simulated yaw rate

Table 1 displays the average error between measured vehicle data and simulated data and it shows that the dynamics of the simulation are realistic enough to test different control techniques.

Average relative error		
v_x	a_y	$\dot{\psi}$
1.4%	3.4%	1.7%

Table 1: Average error between vehicle data and simulation

3. Artificial Neural Networks Model

The goal in this section is to identify the vehicle model with an artificial neural network, using an adequate network architecture and a set of parameters which best model the vehicle system, described by an appropriate set of input-output data.

3.1. Data

The inputs of the neural network are the steering angle (δ) and the reference longitudinal velocity (v_{ref}). The outputs are the velocities of the vehicle (v_x, v_y and $\dot{\psi}$). Feedback exists for both inputs and outputs. This can be formulated as the following NARX model, also illustrated in Figure 5:

$$y(k) = F[\mathbf{u}(k-1), \mathbf{u}(k-2), \mathbf{y}(k-1), \mathbf{y}(k-2)] \quad (2)$$

where k denotes the current sampling instant, the input vector is represented by $\mathbf{u}$ and the output vector is represented by $\mathbf{y}$.

This ANN is to be used as a prediction model for MPC. As such, for each time step, past simulated values from v_x, v_y and $\dot{\psi}$ are available for the the feedback delays in the network, but during the prediction horizon H_p these values are obtained through feedback from the ANN outputs.

Figure 5 shows the inputs and outputs of the neural network with the respective delays.

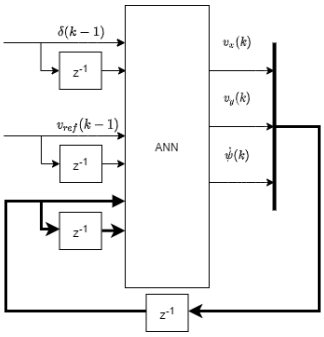

Figure 5: Closed loop neural network structure

The data used to construct the neural network model has to be divided in three different categories: training, validation and testing. The training data subset is used to directly estimate the weights and biases of the ANN, which means that performance estimates relative to the training dataset are biased. The validation dataset is used to rank multiple designs and to determine when overtraining and overfitting begin to occur. Overtraining occurs when the performance on the training data is increased at the expense of deteriorating the performance on the nontraining data. Overfitting occurs when more weights and biases then necessary are used. Validation data is thus used to measure network generalization and to halt training when generalization stops improving. Finally, the testing dataset is used to obtain unbiased estimates of performance on nontraining data [12].

For the selection of the training, validation and testing groups, several data from different tracks was available. All the data was recorded between

July and August of 2019 with FST09e, from the following events: FS East endurance event; FS East skidpad event; FSG endurance event and Stuttgart practice track. As such, the available data was compared between each other in order to provide more insight to which tracks covered the situations the dynamical model can experience. Training is done with approximately two laps of FS East endurance, validation is done with approximately 1 lap of FSG endurance and the test is done with a skidpad run (2 circles to the right and 2 circles to the left).

3.2. Data Preprocessing

Since the steering encoder and the used GPS have different sampling times (0.02s and 0.055s respectively), the data from these sensors was resampled with a sampling time of 0.055s.

Neural networks models learn a mapping between input and output variables. As such, the scale and distribution of the data may be different for each variable. To ensure all variables are learned, the data was rescaled using standardization, which rescales the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. A value $u(k)$ is standardized to $u(k)_{stdd}$ by the following equation:

$$u(k)_{stdd} = \frac{u(k) - \overline{u}}{\sigma_u} \tag{3}$$

where $\overline{u}$ represents the sample mean and σ_u represents the standard deviation of the sample.

3.3. Application and Results

Supervised batch learning was used since all the data of the inputs and target outputs is available *a priori*. The learning algorithm used was the Levenberg-Marquardt. The performance of the ANN is measured by the mean squared error (MSE), which is the sum of the difference between the real target outputs ($\mathbf{y}(k)$) and the ones calculated by the ANN ($\hat{\mathbf{y}}(k)$), divided by the total number of samples N, according to Equation 4:

$$MSE = \frac{1}{N}\sum_{k=0}^{i=N}(\mathbf{y}(k) - \hat{\mathbf{y}}(k))^2 \tag{4}$$

Training is performed in two stages. In the first stage the network is created and trained in open loop form, as shown in Figure 6. This allows the network to be supplied with the correct past outputs during training to produce the correct current outputs. Afterwards, the network is converted to closed loop, as in Figure 7, which is the way it is intended to be used, and it is retrained in closed loop, to further improve its performance, using the network trained in open loop as a starting point. In both figures it can be seen that the hidden layer activation function is a sigmoid while the output layer activation function is linear.

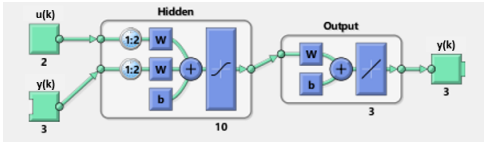

Figure 6: Neural network structure in open loop during training

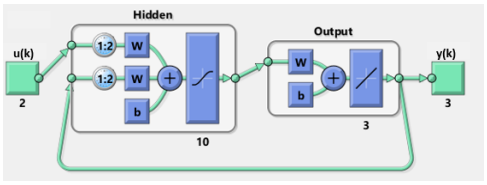

Figure 7: Neural network structure in closed loop

Several parameters were varied and the best results were obtained with one hidden layer with 10 neurons, with 2 delays for both input and output feedback.

In Figures 8, 9 and 10 the results in closed loop are presented, i.e. the neural network used its own outputs as feedback, and not the target outputs. It can be seen that good approximations were achieved on all datasets. This indicates that the obtained neural network may be used as a reliable dynamical model for the vehicle. Training, validation and testing are shown continuously but during training, each dataset had its separate initialization values on the output feedback.

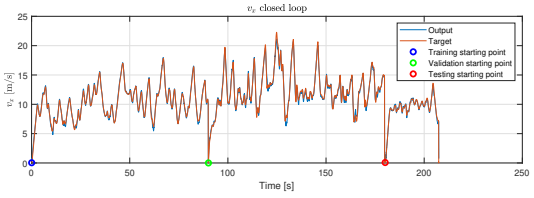

Figure 8: Comparison between the target v_x and the output of the ANN

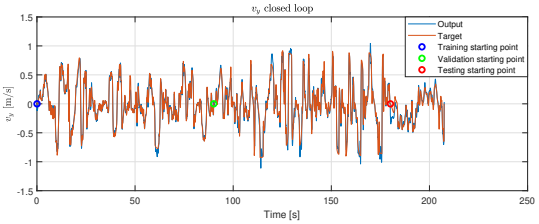

Figure 9: Comparison between the target v_y and the output of the ANN

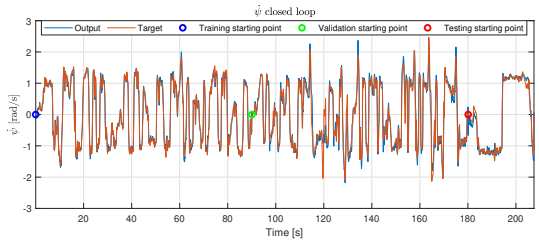

Figure 10: Comparison between the target $\dot{\psi}$ and the output of the ANN

In Figure 11 the MSE is shown with respect to the number of iterations. The test curve shows that the neural network performed well in the skidpad, a track with a different layout.

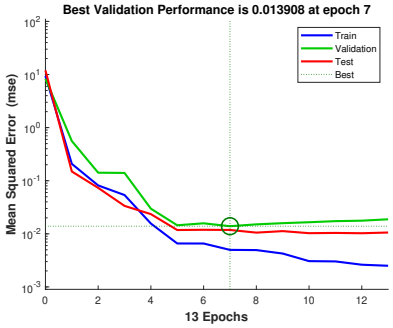

Figure 11: Performance of the ANN

4. Nonlinear Model Predictive Control

4.1. Contouring Formulation

The objective of the contouring formulation is to follow a reference path as fast as possible. The contouring formulation from [13] is adapted and implemented in the present case. The reference paths are the trajectories obtained in the vehicle model shown in Section 2, using as inputs real data from steering angle and reference velocity. The reference path is then parameterized by arc length (t). For this, a third order spline is used, since it offers a fast way to evaluate any point along the contour $(X_{ref}(t), Y_{ref}(t))$. In order to follow the path, the position of the car (X, Y) has to be connected to the position on the path, i.e. the arc length. This arc length is represented by t and it can be computed by projecting the position of the car (X, Y) onto the reference path.

The arc length parameter is represented by $t \in [0, L]$, where L is the total length. The splines used for this parameterization are obtained by an offline fitting of the trajectory. Using this parameterization, any point $X_{ref}(t), Y_{ref}(t)$ on the path can be obtained by evaluating a third order polynomial for its argument t. The angle of the tangent to the path at the reference point, $\Phi(t)$, with respect to the x-axis can be calculated by:

$$\Phi(t) \triangleq arctan\Big(\frac{\partial Y_{ref}(t)}{\partial X_{ref}(t)}\Big) \tag{5}$$

Error measures define the deviation of the current position of the car X, Y from the desired reference point $X_{ref}(t), Y_{ref}(t)$ and this measure is needed to formulate the MPC problem. This deviation is the contouring error, which can be visualized in Figure 12:

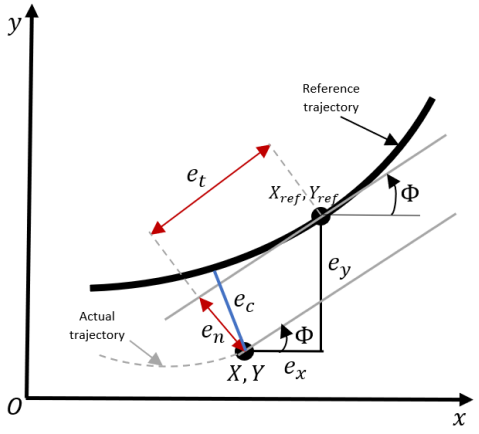

Figure 12: Contouring error of the vehicle (X, Y) relative to the reference trajectory (X_{ref}, Y_{ref})

Let $t_{min} : \mathbb{R}^2 \longrightarrow [0, L]$ be a projection operator on the reference trajectory defined by:

$$t_{min} \triangleq arg\min_t (X - X_{ref}t))^2 + (Y - Y_{ref}(t))^2 \tag{6}$$

where t_{min} is the arc length that corresponds to the closest point in the reference path in relation to the current car position. The orthogonal distance from the car to the reference path is given by the contouring error e_c, which can be approximated by its normal component e_n:

$$e_n(X, Y, t_{min}) \triangleq sin(\Phi(t_{min})) * (X - X_{ref}(t_{min})) - \\ - cos(\Phi(t_{min})) * (Y - Y_{ref}(t_{min})) \tag{7}$$

Note that this is not the same as calculating the Euclidean distance because of the tracking error, represented in Figure 12 by e_t, which inevitably exists because the values of t are discretized. The presented formulation remains accurate even if tracking error is present.

4.2. MPC Problem Formulation

Figure 13 shows the overview of the proposed control architecture. Since all simulations are to be made in Simulink, the nonlinear MPC block is used,

customizing it to use the cost function in Equation 8a with the respective constraints and the ANN state prediction model described in Section 3.3. The function *fmincon* is used to minimize the cost function while respecting the constraints. The MPC outputs the two control actions: steering angle (δ) and reference velocity (v_{ref}), which are the inputs given to the car dynamical model detailed in Section 2. This model then outputs the velocities v_x, v_y and $\dot{\psi}$. which are converted to global coordinates and feedback to the MPC. The time step (T_s) of the MPC is the same as the ANN: 0.055s.

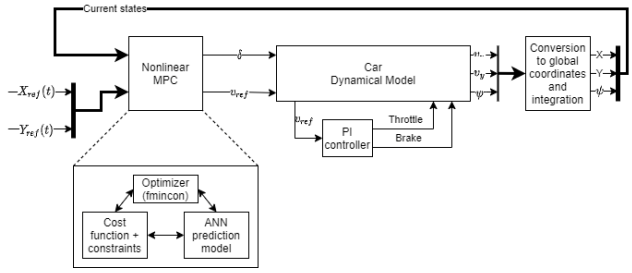

Figure 13: Proposed Nonlinear MPC Control Architecture

4.3. Cost Function

After defining the error measures, the MPC problem can now be formulated. The formulation shown is based on the one used in [13] with some adaptations. The goal is to minimize the contouring error while maximizing the progress along the track over a finite horizon of H_p sampling times, while respecting model dynamics and input constraints:

$$J = \min \sum_{k=1}^{H_p} ||e_n(k)(X(k), Y(k), t_{min}(k))||^2 w_n - \\ - w_t * t_{min,N} + ||\Delta\delta(k)||^2 w_{\Delta\delta} + \\ + ||\Delta v_{ref}(k)||^2 w_{\Delta v_{ref}} + ||\delta|| w_\delta \quad (8a)$$

$$s.t. \quad \mathbf{y}(k) = F[\mathbf{y}(k-1), \mathbf{y}(k-2), \mathbf{u}(k-1), \mathbf{u}(k-2)] \quad (8b)$$

$$\underline{\mathbf{u}} \leq \mathbf{u}(k) \leq \overline{\mathbf{u}} \quad (8c)$$

$$\underline{\Delta\mathbf{u}} \leq \Delta\mathbf{u}(k) \leq \overline{\Delta\mathbf{u}} \quad (8d)$$

where $X(k), Y(k)$ is the position of the car at time step k, determined by the nonlinear prediction model $F[.]$ in Equation (8b), which is based on the ANN obtained in Section 3.3, where $\mathbf{u}(k)$ represents the input vector and $\mathbf{y}(k)$ represents the output vector. The term e_n is the approximation to the contour error defined in Equation (7). By subtracting the term $t_{min,N}$ to the cost function, the arc length parameter t is maximized, thus the car progress over the track is also maximized. The variable δ is the steering angle control action, and v_{ref} is the reference velocity control action. This cost function minimizes the variations of the control actions, and it also minimizes the module of the steering wheel angle. Tuning weights (w_i) exist for each parameter.

Constraints (8c) and (8d) limit the inputs $\mathbf{u}$ to physically admissible values. These constraints limit the rotation of the steering wheel and its maximum rotation speed. The maximum rotation depends on the steering geometry and the maximum rotation speed depends on the actuators used for the steering wheel in the autonomous vehicle. A minimum vehicle velocity of 5 km/h was defined to avoid trivial solutions and a maximum velocity of 30 km/h was defined. This value can be increased once the algorithm is successfully tested in the real prototype. The variable Δv_{ref} represents the longitudinal acceleration of the car and the upper limit for the constraint is the maximum longitudinal acceleration of the car obtained during simulations, while the lower limit was set considerably lower than the maximum negative acceleration simulated, which was 19.4 m/s^2. This is to prevent solutions with the car braking on the limit of tyre grip while also turning the steering wheel, which leads to unstable behaviour. A human driver has the ability to brake and turn at the same time while balancing the limits of grip, but this is harder to reproduce with a controller.

4.4. Simulation Results

In order to test the developed controller, a reference trajectory is given. For this, data from FS East endurance is used. The steering angle and reference velocity are given to the dynamical model and the obtained output trajectory is saved and used as reference for the MPC. In Figure 14 the trajectory obtained with this simulation is shown. The blue markers show the track limits. In Formula Student competitions the track width varies along the track but the minimum width is 3m, according to competition rules [14]. To take in consideration the fact that the trajectories shown are for the CoG of the car, the track limits represented correspond to the minimum track width subtracting the width of the car ($1.2m$): $3 - 1.2 = 1.8m$. Hence if the output trajectory is within the limits of the blue markers, this means that the whole car should stay inside the track.

The sampling time of the controller is $T_s = 0.055s$ and the prediction horizon length is $H_p = 15$ time steps, which gives an ahead prediction of 0.825s. The control horizon is $H_c = 3$ time steps. Several combinations of these parameters were used but the closest tracking of the reference trajectory was achieved with this set of parameters.

It can be verified that the maximum velocity of

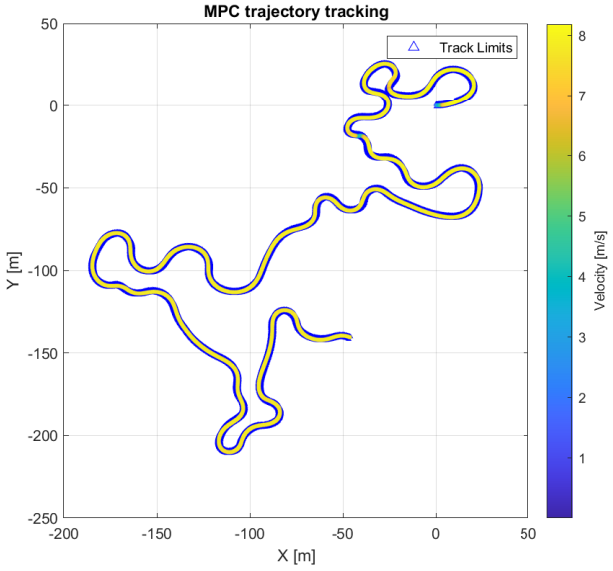

Figure 14: Simulated trajectory with MPC with the velocity profile for FS East track

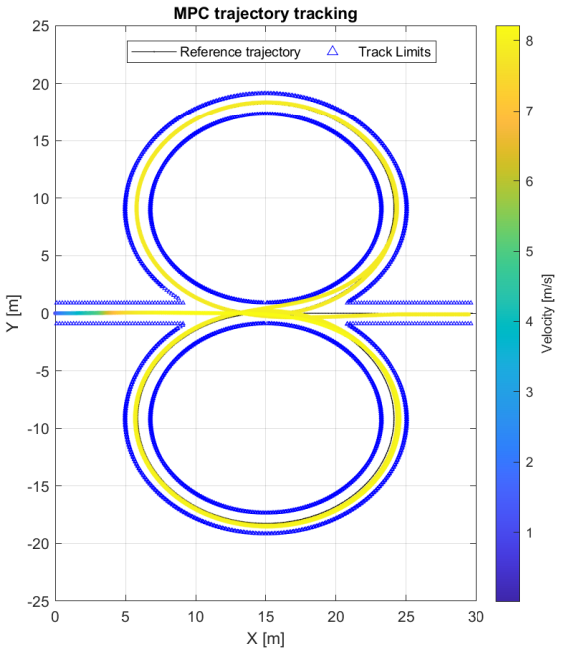

Figure 15: Simulated trajectory with MPC with the velocity profile for skidpad track - the car starts on the left, performs two circles to the right followed by two circles to the left and finishes in the middle

the car was constrained to 30 km/h, which is approximately 8.3 m/s. The color of the trajectory indicates the velocity of the car, which is constant throughout most of track, as indicated by the yellow line in Figures 14 and 15. The car stays inside the track limits throughout the entire run.

Figure 15 shows the MPC controller following the reference trajectory of the centreline of the skidpad. From this Figure, the car also stays within track limits during the entire simulation. The parameters used for the skidpad are the same as for FS East, except for one change that was done due to the fact that the track intersects itself, which only happens in the skidpad event. Previously, when calculating the contouring error, the argument t_{min} was used, which is the arc length that corresponds to the reference point closest to the current trajectory point. This is valid when the track does not intersect itself, but if that is not the case, then the closest position point may not be the right one to use as a reference. To solve this issue for the case of the skidpad, the references for the calculation of the contouring error are calculated with the estimation $t_{min}(k+1) = t_{min}(k) + v_x \times T_s$. This means the reference trajectory over the prediction horizon, instead of corresponding to the points closest to the current trajectory, corresponds to the points $X(t), Y(t)$ with the parameter t closest to t_{min}.

4.5. Runtime

Due to the complexity involved in the nonlinear optimization and in the nonlinear model used, the computation time is longer than the sampling time of 0.055s by a very significant amount, thus it is not possible to implement this controller in the real prototype with the current technology. Even reducing the prediction and control horizons and increasing the sampling time proved to not be enough. The computer used to run this simulation has an Intel® Core™ i7 5700HQ processor, which has similar capabilities to the hardware currently used in the car for its control algorithms.

5. Learning the MPC with an ANN

5.1. ANN Structure and Training

Due to the long computation times of the controller, a different implementation approach is tested. In Section 3.3 it was seen how a neural network could learn the dynamics of the vehicle (v_x, v_y and $\dot{\psi}$) that result from inputs of steering angle (δ) and reference velocity (v_{ref}). If instead, the neural network learns what the steering angle and the reference velocity should be for a certain progression of reference trajectory coordinates over time $(X_{ref}(k), Y_{ref}(k))$, then it may be possible to control the vehicle using such network without the need for online computation of the control actions. Two ANN model structures are going to be analysed in this chapter: NARX and NFIR models. Unlike a NARX model, which requires output feedback, a NFIR model does not. In the neural network controller proposed in this chapter, there is no way

to guarantee that the outputs of the ANN to be trained will be sufficiently accurate. If the neural network receives wrong outputs through feedback, the errors will propagate. This is a more exigent situation than when the ANN is used as a prediction model in the MPC, since in that case the model predictions are feedback only for a limited H_p horizon.

Since the trajectory is previously known but the velocity is not, there is no information about the sampling time of each input for training the ANN. Instead of spacing the data with equal sample times, the data was spaced with equal sample distances, which means that between each sample, the car covers 2 cm.

Using third order polynomials just like in Section 4.1, it is possible to parameterize the track by its arc length, which is equivalent to the distance traveled along the track. The track was parameterized with small equal arc length distances, and for each data sample, the steering angle and the reference velocity were interpolated from simulation data from the car dynamical model (Section 2). The data used for learning was also standardized according to the method described in Section 3.2.

To try to improve the learning results, additional data preprocessing is done. The coordinates of the reference trajectory X_{ref}, Y_{ref} are inputs of the ANN. These coordinates can be converted to coordinate variations $(\Delta X_{car}, \Delta Y_{car})$ on the vehicle reference axis, which should prevent the ANN from learning only specific track coordinates. For this, a similar formulation to the trajectory error defined in Section 4.1 is used, as shown in Figure 16 with two consecutive points of a given trajectory. The variable ΔY_{car} can be seen as the normal reference trajectory variation, relative to a line tangent to the reference point at instant k while ΔX_{car} can be seen as the parallel reference trajectory variation along the tangent to the reference point at instant k, as demonstrated by Equations 10 and 11, being Φ the angle of the tangent relative to the x-axis:

$$\Phi(k) \triangleq arctan\left(\frac{\partial Y_{ref}(k)(t)}{\partial X_{ref}(k)(t)}\right) \tag{9}$$

$$\Delta X_{car} \triangleq -cos(\Phi(k)) * \Big(X_{ref}(k) - X_{ref}(k+1)\Big) - \\ - sin(\Phi(k)) * \Big(Y_{ref}(k) - Y_{ref}(k+1)\Big) \tag{10}$$

$$\Delta Y_{car} \triangleq sin(\Phi(k)) * \Big(X_{ref}(k) - X_{ref}(k+1)\Big) - \\ - cos(\Phi(k)) * \Big(Y_{ref}(k) - Y_{ref}(k+1)\Big) \tag{11}$$

Since the MPC has information about future states when computing the control actions, the same type of information may be useful for the ANN to better learn the control actions. Therefore, additional inputs are given to the ANN when training, as will be shown in Figures 17 and 18. These inputs correspond to future reference position variations, so that at instant k the ANN receives information about the current and future position variations, which is more similar to how an MPC controller works. Different values for the number of future reference input signals, n, were experimented but the best results were achieved with $n = 11$. This value is similar to the prediction horizon of the MPC, $H_p = 15$.

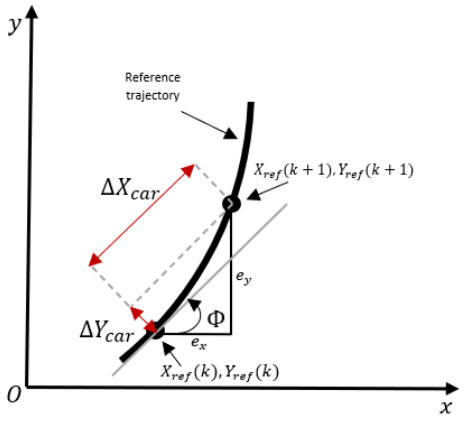

Figure 16: Coordinate variations ΔX_{car} and ΔY_{car} relative to vehicle reference frame at instant k

Two main types of neural network architecture are tested: ANN controller 1 - receives as inputs only the coordinates variations of the reference trajectory $(\Delta X_{car}, \Delta Y_{car})$, and ANN controller 2 - which also receives as inputs the current states of the vehicle (X,Y and ψ). A scheme of these ANN is displayed in Figures 17 and 18, where the NFIR model with one input delay is represented for simplicity. Adding the states as inputs makes the ANN controller similar to an MPC controller, which also receives the current states at each time step. For the ANN controller 1, without the vehicle states, the training was done with data from the real prototype on FS East endurance and FSG endurance, while for the ANN controller 2, with the vehicle states, the data used was from simulation with the MPC controller on FS East endurance and FSG endurance as well.

To determine the best architecture for each of the two ANN controllers, different ANN are trained and compared by varying the number of delays, input delays in the case of NFIR models and both input and feedback delays in the case of NARX models, and the number of future reference input signals.

The lowest MSE for ANN controller 1 is achieved

with a NFIR model with 2 input delays and 11 future reference input signals, shown in Figure 17, while the lowest MSE for ANN controller 2 is achieved with a NFIR model with 1 input delay and 11 future reference input signals, as shown in Figure 18.

Several numbers of hidden layers and neurons were experimented and best results were achieved with 2 hidden layers, with 25 neurons each. For ANN controller 1 the MSE on the training data is 1.02, while for ANN controller 2, the MSE achieved is better: 0.25.

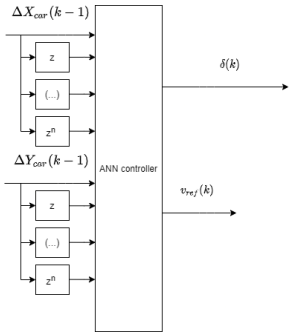

Figure 17: ANN controller 1 with future reference input signals

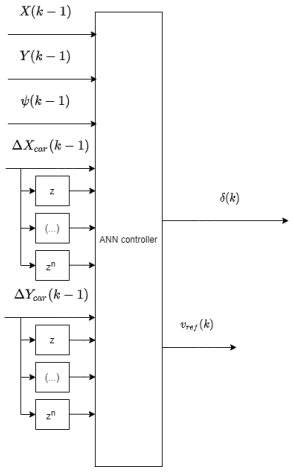

Figure 18: ANN controller 2 with future reference input signals

5.2. Results

Both ANN were simulated, replacing the MPC controller in the closed-loop configuration. The trajectory obtained for FS East endurance track with ANN controller 1 is shown in Figure 19. During this simulation, the reference trajectory is compared to the simulated one, and the control action corresponding to the closest reference point is given. This controller tracks the reference trajectory, represented with a black line, for some corners until a certain time instant. Even though the MSE between the outputs and the targets of the ANN is relatively small, the accumulation of errors ends up being significant. This brings out the major issue with this type of controller, which is its inability to feedback errors, unlike MPC or even PID controllers.

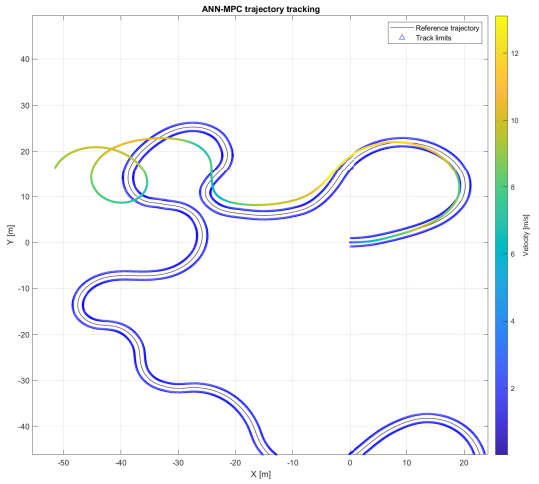

Figure 19: Simulated trajectory with ANN controller on FS East track

The trajectory obtained with the ANN controller 2 is shown in Figure 20. This controller starts the corner well, but quickly goes offtrack. This may be due to the fact that the data given for the ANN to learn contained states (X, Y and ψ) of a vehicle that followed the reference trajectory closely, as shown in Figure 14. When the vehicle follows the trajectory almost perfectly, there is not enough dynamical information for the ANN to learn well.

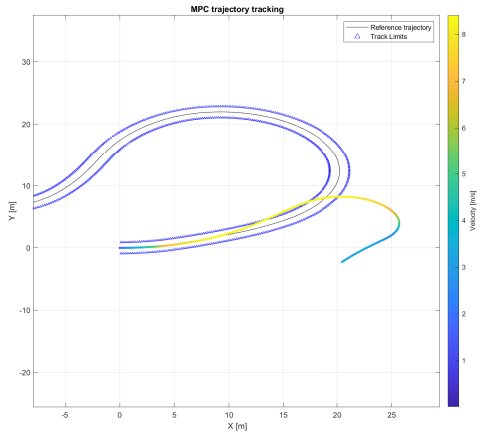

Figure 20: Simulated trajectory with ANN controller 2 on FS East track

6. Conclusions

Using the principles of vehicle dynamics, a car model was obtained and validated by comparing its results to real data. Data from testing and from competitions was selected and preprocessed in order to train a neural network that is a good approximation to the vehicle model. In order to design an MPC, the reference trajectory was parameterized with respect to its arc length and the contour error was defined. The cost function is such that the contouring error is minimized while progress along the track is maximized.

It was proven that it is possible to use a prediction model based on an artificial neural network to control the vehicle in a defined trajectory. Despite the advantages ANN provide of not requiring knowledge about the system due to black box modelling, the nonlinear optimization still takes a significant amount of computation time, which is a barrier for real-time implementation of this control technique.

To attempt to solve the long computation times, several different neural network models were tested to learn the model predictive controller dynamics. In spite of the ANN models providing a good approximation of the simulated output controller signals in the learning phase, they were not able to sustain the same performance in a real-time implementation due to the feedback error accumulation.

After good results and reasonable computation times are obtained in simulation, one of the following steps is to implement the control algorithm in a real Formula Student prototype. Other control strategies may be explored such as Adaptive MPC [15] and Explicit MPC [16].

References

[1] C.F. Kerry and J. Karsten. Gauging investment in self-driving cars, 2017. [Online; accessed on August 2020].

[2] J. Kabzan, M.I. Valls, V.J.F. Reijgwart, H.F.C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart. AMZ Driverless: The Full Autonomous Racing System. May 2019.

[3] R Shreyas, A. Bradley, and G. Collier. MPC Controller for Autonomous Formula Student Vehicle. In *SAE Technical Paper*. SAE International, March 2020.

[4] J. Richalet, A. Rault, J. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14:413–428, September 1978.

[5] M. Arnold and G. Andersson. Model Predictive Control of energy storage including uncertain forecasts. In *Proceedings of the 17th Power Systems Computation Conference (PSCC 2011)*, Stockholm, Sweden, August 2011.

[6] T. Geyer. *Model predictive control of high power converters and industrial drives*. Wiley, 2016. ISBN:978-1-119-01090-6.

[7] E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer, 1999.

[8] M. Lazar and O. Pastravanu. A neural predictive controller for non-linear systems. *Mathematics and Computers in Simulation*, 60:315–324, 09 2002.

[9] J. Qin and T. Badgwell. A Survey of Industrial Model Predictive Control Technology. *Control engineering practice*, 11:733–764, 07 2003.

[10] Luís Miguel Marcos Abrunhosa Vieira Abreu. *Mechanical design of the wheel assembly of an electric Formula Student prototype*. Mechanical engineering master's thesis, Instituto Superior Técnico, 2019.

[11] André Manuel Pinheiro Antunes. *Sideslip Estimation of Formula Student Prototype*. Mechanical engineering master's thesis, Instituto Superior Técnico, 2017.

[12] H. Demuth, M. Beale, and M. Hagan. *Neural Network Toolbox^{TM} User's Guide*. The MathWorks, Inc., 2017.

[13] A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, April 2014.

[14] Formula Student Germany. *Formula Student Rules 2020*, 2020.

[15] M. Bujarbaruah, X. Zhang, E. Tseng, and F. Borrelli. Adaptive MPC for Autonomous Lane Keeping. In *14th International Symposium on Advanced Vehicle Control (AVEC)*, July 2018.

[16] C.F. Lee, C. Manzie, and C. Line. Explicit Nonlinear MPC of an Automotive Electromechanical Brake. In *Proceedings of the 17th World Congress*. The International Federation of Automatic Control, July 2008.