IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 May 2022, accepted 21 June 2022, date of publication 24 June 2022, date of current version 30 June 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3186020

APPLIED RESEARCH

Processor-in-the-Loop Validation of a Gradient Descent-Based Model Predictive Control for Assisted Driving and Obstacles Avoidance Applications

PIERPAOLO DINI AND SERGIO SAPONARA, (Senior Member, IEEE)
Department of Information Engineering, University of Pisa, Pisa, 56100 Tuscany, Italy
Corresponding author: Sergio Saponara (sergio.saponara@unipi.it)

This work was supported by the Ministero Istruzione Università Ricerca (MIUR)-Dipartimenti di Eccellenza Project Crosslab.

ABSTRACT For safety-critical applications, the validation process using a model-based approach plays an increasingly important role. In this paper we propose the application of a predictive control algorithm, entirely implemented in low-level code, to the use case of assisted driving of four-wheel vehicles. The aim is to present the workflow for the validation of an advanced control algorithm and its implementation on an Embedded system, representative of the computational capabilities of Automotive ECUs. The proposed validation exploits the SIL (Software-In-the-Loop) and PIL (Processor-In-the-Loop) paradigms to analyse the combination of control parameters and factors related to the choice of the mathematical model describing the vehicle behaviour and the choice of the numerical algorithms selected to approximate the differential equations.

INDEX TERMS Model predictive control, assistance driving, vehicle dynamics, model-based design, simulation.

I. INTRODUCTION

A. OVERVIEW ON ASSISTED DRIVING

An autonomous vehicle is able to detect its environment and navigate with partial or even null driver action [1]. Information from different perception techniques of the surrounding environment are usually combined, e.g. radar, lidar, GPS, odometry and computer vision.

An advanced control algorithm interprets the sensory information to identify appropriate navigation paths, as well as obstacles and signs. The functionalities required of an autonomous vehicle are Perception, Localization, Planning, Vehicle Control and System Management.

This paper is focused on vehicle trajectory control implemented through a predictive control algorithm, assuming that the necessary sensors are present on board the vehicle to provide information about the surrounding environment.

In the first hierarchical block a route is planned based on the road network. The road network is described by means of a directed graph, with weights corresponding to the costs of crossing road segments, so that this process can be described as the problem of finding a minimum cost route on a graph (operational search).

The second hierarchical level is called ''behavioural'', where the driving algorithm solves locally the problem of how to move the vehicle forward, respecting the road lanes. The third hierarchical level concerns the planning of the vehicle's motion by calculating a trajectory on which the vehicle must move in order to perform a local navigation task. Finally, the fourth hierarchical level is the control system that reactively corrects errors in the execution of the motion planned in the previous level [2].

As anticipated in this article we deal with vehicle control which requires the presence of the trajectory planning system. With reference to the the classification of 6 autonomy levels, from L0 (no assistance) to L5 (fully autonomous), published by the SAE (Society of Automotive Engineers), in this work we propose the use of Model Predictive Control (MPC) to implement an assisted driving algorithm capable of processing an obstacle avoidance manoeuvre, so ensuring L3 and being at the core of emerging L4 and L5 vehicles.

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

The challenges are to design control and sensory systems that are so accurate in processing the vehicle's road travel information that they ensure maximum passenger safety and provide vehicle comfort and low fuel consumption.

B. STATE OF THE ART

Modern research on self-driving cars generally uses Bayesian simultaneous localization and mapping (SLAM) algorithms, which fuse data from multiple sensors and an off-line map into current location estimates and map updates [3].

Researchers are developing a variant of SLAM, with detection and tracking of other moving objects (DATMO), which also handles obstacles such as cars and pedestrians [4], [5].

Simpler systems may use roadside real-time locating system (RTLS) technologies to aid localization. Typical sensors include Lidar, stereo vision, GPS and IMU. Udacity is developing an open-source software stack [6].

Control systems on autonomous cars may use Sensor Fusion, which is an approach that integrates information from a variety of sensors on the car to produce a more consistent, accurate, and useful view of the environment [7].

Being highly flexible, MPC (Model Predictive Control) started to be implemented also in automotive field for active safety purposes. It can manage in fact not only the path planning problem but also threat assessment and hazard avoidance.

As example, [8] assumes that road lane data is available and that road hazards have been detected, located, and mapped into a 2-dimensional corridor of travel. This is possible thanks to a proper system of sensors, such as LIDAR, radar and camera and an optimal sensor fusion.

These sensors should provide lane, position and environmental information needed for the application. Thanks to these data we can obtain constrained vectors in the prediction horizon, in order to evaluate the minimum threat pose.

Another interesting application was in [9], where they account for the uncertainty in the traffic environment by a small number of future scenarios, which is intuitive and computationally efficient. These scenarios can be generated by any model-based or data-based approach, resulting in a good performance for highways' scenarios.

The artificial potential field can be an optimal method to deal with different types of constraints. An application using the MPC and the APF (Artificial Potential Functions) is done in [10]–[12].

The main idea behind is to associate at every constraint a proper potential field, so that the minimum values are in the center of the vehicle's lane. Furthermore the MPC can manage also low friction road conditions such as demonstrated in [13].

In order to deal with these limit conditions, they take into account highly nonlinear models, which consider both wheel dynamic and load transfer, resulting in a high computational request. For this reason an auto-generated tailored NLMPC (Non Linear MPC) is implemented thanks to the ACADO Code Generation tool [14].

The MPC technique has been implemented not only for active safety systems but also for vehicle dynamics, driver modeling and integrated chassis control systems [15], [16].

Nonetheless the major focus is in active safety with many possible applications such as active steering [17], [18], active braking [19], active traction [20] and active differentials or suspension to coordinate and improve the vehicle handling [21], the stability and the ride comfort while avoiding collisions [22].

In [23], authors propose an interesting NLMPC with improved computationally efficiency, followed by detailed validation tests in simulation.

However, the limit of the above mentioned state-of-art techniques for trajectory planning and obstacle avoidance is that the proposed techniques are usually modelled at high level and require a computational complexity not suitable for real-time implementation in the resources-constrained embedded systems that typically are present in automotive ECUs.

C. CONTRIBUTIONS

To overcome this limit, the contribution of this work is to present an integrated algorithm capable of calculating vehicle trajectories and avoid obstacles, while generating the necessary control actions, with a complexity suitable for real-time implementation in automotive embedded systems.

This work is focused on the third and fourth levels of the decision hierarchy in Section I.A, assuming that the vehicle has the necessary technology on board to implement the first two hierarchical levels.

Our implementation makes use of the GRAMPC (GRadient-based Augmented Lagrangian MPC) library, which implements at low-level a variant of MPC where the numerical optimiser exploits gradient descent, often used also in Adaptive [24], [25] and Learning-based [26], [27] Control Systems for Mechatronics, and is implemented entirely in the C/C++ language.

Another contribution of our proposed work is the validation of the control algorithm with a Processor-in-The-Loop paradigm, together with an exhaustive analysis of the computational cost associated with the choice of used mathematical models and the choice of numerical methods and parameters in the definition of the constrained optimisation problem.

As a further contribution, we propose the analysis of robustness to measurement noise, emulated in the context of PIL validation, exploiting a ZCU104 that has a real-time processor on which the control algorithm is run, and has an FPGA that is used as a HW accelerator to emulate the measurement noise.

We can summarize main claims as: (i) implementation in C/C++ language and verification of an NLMPC-based assisted driving algorithm and gradient descent algorithm-based optimizer, to lighten the computational load; (ii) in-depth SIL and PIL validation of GRAMPC algorithm and embedded implementation performance; (iii) PIL verification on core Embedded Cortex-A/R + FPGA systems;

(iv) real-time simulation of noise injection on FPGA, for closed-loop robust stability analysis; (v) analysis of computation time and performance as a function of variation of the mathematical model representing the vehicle kinematics/dynamics; (vi) analysis of computation time and performance as a function of variation in the method of numerical approximation of differential equations.

D. PAPER ORGANIZATION

The rest of the paper is organized as follows: Section II presents the GRAMPC Framework in terms of theoretical introduction on the implemented NLMPC and library description; Section III introduce the kinematics, dynamics and augmented models for vehicle representation; Section IV reports the SIL validation with several configuration of GRAMPC parameters; Section V show further analysis for validate the GRAMPC solution in terms of computational time analysis varying vehicle model and numerical approximation method, and further robustness analysis in terms of measurement noise injection and model uncertainty; Conclusions and considerations on future development are reported in Section VI.

II. DESCRIPTION OF THE GRAMPC FRAMEWORK

A. THE GRAMPC ALGORITHM

A constrained optimal control problem (OCP) is solved in GRAMPC, as reported in Eq.1.

$$\begin{aligned}
\min_{u,p,T}\ & J\left(u,p,T,x_0\right)=V\left(x_T,p,T\right)\\
&+\int_0^T L(x(t),u(t),p,t)dt\\
&\dot{x}(t)=f(x(t),u(t),p,t),x\left(t_0\right)=x_0\\
&g(x(t),u(t),p,t)=0\\
&g_T\left(x_T,p,T\right)=0\\
&h(x(t),u(t),p,t)\leq 0\\
&h_T\left(x_T,p,T\right)\leq 0\\
&u(t)\in[u_{\min},u_{\max}]\\
&p\in[p_{\min},p_{\max}]\\
&T\in[T_{\min},T_{\max}]
\end{aligned}\tag{1}$$

With the following symbolic meaning: $u \in \mathrm{R}^{N_u}$ is the control vector, $p \in \mathrm{R}^{N_p}$ is the parameter vector, $x \in \mathrm{R}^{N_x}$, T is the prediction time horizon, $f(x,u,p,t)$ is the vector field describing the dynamics of the system, $g(x,u,p,t)$, $g_T(x_T,p,T)$ represent the set of equality alleys, $h(x,u,p,t)$, $h_T(x_T,p,T)$ represent the set of inequality constraints, and finally $\Omega_u = [u_{min},u_{max}]$, $\Omega_p = [p_{min},p_{max}]$, $\Omega_T = [T_{min},T_{max}]$ define the operational constraints.

GRAMPC functions implement an augmented Lagrangian formulation based on the penalty method, where the gradient descent algorithm is exploited to optimize the iterative update of the variables u, p and T (if enabled).

For each set of constraints in OCP we define a Lagrange multiplier and a penalty factor (in general these are vectors of dimension equal to the set of alleys).

$$\begin{aligned}
\bar{\mu}&=\left[\mu_g,\mu_{gT},\mu_h,\mu_{hT}\right]\\
\bar{c}&=\left[c_g,c_{gT},c_h,c_{hT}\right]
\end{aligned}\tag{2}$$

Eq. 2 returns the vectors in question for the augmented formulation, which allow us to re-write the sets of constraints (Eq. 3), making OCP an equivalent unconstrained optimization problem.

$$\begin{aligned}
&\bar{g}\left(x,u,p,t,\mu_h,c_h\right)\\
&\quad=\begin{bmatrix} g(x,u,p,t)\\ \bar{h}\left(x,u,p,t,\mu_h,c_h\right)\end{bmatrix}\\
&\quad\bar{h}=\max\left\{h(x,p,u,t),-\operatorname{diag}\left(c_h\right)^{-1}\mu_h\right\}\\
&\bar{g}_T\left(x_T,p,T,\mu_{h_T},c_{h_T}\right)\\
&\quad=\begin{bmatrix} g_T\left(x_T,p,T\right)\\ \bar{h}_T\left(x_T,p,T,\mu_{h_T},c_{h_T}\right)\end{bmatrix}\\
&\quad\bar{h}_T=\max\left\{h_T\left(x_T,p,T\right),-\operatorname{diag}\left(c_{h_T}\right)^{-1}\mu_{h_T}\right\}
\end{aligned}\tag{3}$$

Through Eq.3, it is possible to re-write the terms of the cost function as given in Eq.4.

$$\begin{aligned}
\tilde{J}\left(x,u,p,T,\bar{\mu},\bar{c},x_0\right)&=\tilde{V}\left(x_T,p,T,\mu_T,c_T\right)\\
&\quad+\int_0^T\check{L}(x,u,p,t,\mu,c)dt\\
\tilde{V}\left(x_T,p,T,\mu_T,c_T\right)&=V\left(x_T,p,T\right)\\
&\quad+\mu_T^T\bar{g}_T\left(x_T,p,T,\mu_{h_T},c_{h_T}\right)\\
&\quad+\frac{1}{2}\left\|\bar{g}_T\left(x_T,p,T,\mu_{h_T},c_{h_T}\right)\right\|_{c_T}^2\\
\tilde{L}(x,u,p,t,\bar{\mu},\bar{c})&=L(x,u,p,t)\\
&\quad+\mu^T\bar{g}\left(x,u,p,t,\mu_h,c_h\right)\\
&\quad+\frac{1}{2}\left\|\bar{g}\left(x,u,p,t,\mu_h,c_h\right)\right\|_c^2
\end{aligned}\tag{4}$$

For convenience we used the following notation $\mu_T^T = \left[\mu_{gT}^T,\mu_{h_T}^T\right]$; $c_T^T = \left[c_{gT}^T,c_{h_T}^T\right]$; $\mu^T = \left[\mu_g^T,\mu_h^T\right]$ and $c^T=\left[c_g^T,c_h^T\right]$.

In this way it is possible to redefine OCP as an unconstrained optimization, and in particular as a MAX-MIN type optimization problem, as reported in Eq. 5.

$$\begin{aligned}
&\max_{\bar{\mu}}\left\{\min_{u,p,T}\tilde{J}\left(x,u,p,T,\bar{\mu},\bar{c},x_0\right)\right\}\\
&\text{s.t}\,\dot{x}=f(x,u,p,t),x_0\\
&\quad u\in[u_{\min},u_{\max}]\\
&\quad p\in[p_{\min},p_{\max}]\\
&\quad T\in[T_{\min},T_{\max}]
\end{aligned}\tag{5}$$

In GRAMPC the inner minimization problem it is solved through the gradient descent approach, while the external maximization problem it is solved with the steepest ascent approach.

The main control function is then divided in the external cycle in which $\bar{\mu}$ and $\bar{c}$ are updated, and the inner cycle in which are updated the variables u,p and, if enabled, T.

At the end of each cycle the convergence criterion it is invoked for check the "quality" of solution provided by GRAMPC.

- Initialize $\bar{\mu}^{(0)}, \bar{c}^{(0)}$ and set tolerances $\varepsilon_g, \varepsilon_{g_T}, \varepsilon_h, \varepsilon_{h_T}, \varepsilon_0$
- ***for*** $i = 1{:}\, i_{max}$ *do*
  - INNER_CYCLE ()
    $\min_{u,p,T} \tilde{J}\left(x^{(i)}, u^{(i)}, p^{(i)}, T^{(i)}, \bar{\mu}^{(i)}, \bar{c}^{(i)}, x_0\right)$
  - Storage constraints values
    $g^{(i)} = g\left(x^{(i)}, u^{(i)}, p^{(i)}, t\right)$
    $g_T^{(i)} = g_T\left(x_T^{(i)}, p^{(i)}, T^{(i)}\right)$
    $\bar{h}^{(i)} = \bar{h}^{(i)}\left(x^{(i)}, u^{(i)}, p^{(i)}, t, \mu_h^{(i)}, c_h^{(i)}\right)$
    $\bar{h}_T^{(i)} = \bar{h}_T{}^{(i)}\left(x_T^{(i)}, p^{(i)}, T^{(i)}, \mu_{h_T}^{(i)}, c_{h_T}^{(i)}\right)$
  - ***If*** CONVERGENCE ()
    break;
    else

    Update multipliers and penalties computing
    $\bar{\mu}^{i+1}$ and $\bar{c}^{i+1}$

end for

List 1. Pseudo routine of the external cycle (max problem).

In the inner cycle it is solved the optimization of the control variables exploiting gradient descent updates for classic unconstrained optimal problems. Defining the Hamiltonian function as following:

$$H(x, u, p, t, \mu, c) = \widetilde{L}(x, u, p, t, \mu, c) + \lambda^T f(x, u, p, t) \tag{6}$$

The gradient-descent algorithm solves iteratively the system of equations in 7.

$$\begin{aligned} \dot{x} &= f(x, u, p, t), x_0 \\ \dot{\lambda} &= -\frac{\partial H(x, u, p, t, \mu, c)}{\partial x} = -H_x(x, u, p, t, \mu, c) \\ \lambda_T &= \frac{\partial \widetilde{V}\left(x_T, p, T, \mu_T, c_T\right)}{\partial x} = \tilde{V}_x\left(x, p, T, \mu_T, c_T\right) \end{aligned} \tag{7}$$

In this way the Pontryagin's principle [28] is satisfied to find the optimal control vector that is compliant with the following optimization sub-problem.

$$\min_{u \in [u_{\min}, u_{\max}]} H(x, u, p, t, \lambda, \mu, c) \quad \forall t \in [0, T] \tag{8}$$

Of course, the result of the inner cycle it is necessary in the new external iteration, setting $u^i = u^{i|j+1}, x^i = x^{i|j+1}, p^i = p^{i|j+1}, T^i = T^{i|j+1}$ and $\eta^i = \eta^{i|j+1}$.

For the updates of the variables $u, p, T, \mu_g, \mu_h, c_g, c_h$ the following relationship are chosen.

$$\mu_g^{i+1} = \begin{cases} \mu_g^i + (1-\rho) c_g^i g^i & \text{if } \left|g^i\right| > \varepsilon_g \; AND \; \eta^i \le \varepsilon_0 \\ \mu_g^i & \text{otherwise} \end{cases}$$

$$\mu_h^{i+1} = \begin{cases} \mu_h^i + (1-\rho) c_h^i \bar{h}^i & \text{if } \left(h^i > \varepsilon_h, \eta^i \le \varepsilon_0\right) \; OR \; \bar{h}^i < 0 \\ \mu_h^i & \text{otherwise} \end{cases}$$

$$c_g^{i+1} = \begin{cases} \beta_1 c_g^i & \text{if } \left|g^i\right| \ge \max\left\{\gamma_1 \left|g^{i-1}\right|, \varepsilon_g\right\} \text{ AND } \eta^i \le \varepsilon_0 \\ \beta_2 c_g^i & \text{if } \left|g^i\right| \le \gamma_2 \varepsilon_g \\ c_g^i & \text{otherwise} \end{cases}$$

$$c_h^{i+1} = \begin{cases} \beta_1 c_h^i & \text{if } \bar{h}^i \ge \max\left\{\gamma_1 \bar{h}^{i-1}, \varepsilon_h\right\} AND \eta^i < \varepsilon_0 \\ \beta_2 c_h^i & \text{if } \bar{h}^i \le \gamma_2 \le \varepsilon_h \\ c_h^i & \text{otherwise} \end{cases}$$

where the coefficients $\varepsilon_g, \varepsilon_{g_T}, \varepsilon_h, \varepsilon_{h_T}$ and ε_0 are the tolerances factors that the user can set or leave as default values; the same for $\gamma_1, \beta_1, \beta_2, \rho$ in the update relationships.

For the update of control variables $u(t)$, parameters $p(t)$, and time $T(t)$, GRAMPC uses the following relationships, based on gradient-descent.

$$\begin{aligned} u^{i+1} &= \text{sat}\left\{u^i - \alpha^i d_u^i, u_{\min}, u_{\max}\right\} \\ p^{i+1} &= \text{sat}\left\{p^i - \gamma_p \alpha^i d_p^i, p_{\min}, p_{\max}\right\} \\ T^{i+1} &= \text{sat}\left\{T^i - \gamma_T \alpha^i d_T^i, T_{\min}, T_{\max}\right\} \end{aligned} \tag{9}$$

Regarding the stability/convergence check, the convergence criterion is defined with the following condition.

$$\left(\begin{array}{c} \begin{bmatrix} \left|g_T^i\right| \\ \max\left(h_T^i, 0\right) \end{bmatrix} \le \begin{bmatrix} \varepsilon_{g_T} \\ \varepsilon_{h_T} \end{bmatrix} \\ AND \\ \max_{t \in [0,T]} \begin{bmatrix} \left|g^i\right| \\ \max\left(h^i, 0\right) \end{bmatrix} \le \begin{bmatrix} \varepsilon_g \\ \varepsilon_h \end{bmatrix} \end{array} \right) OR \; \eta^i \le \varepsilon_0 \tag{10}$$

In any case, the external algorithm terminates when the maximum number of iterations i_{max} is reached, providing a sub-optimal solution. This is necessary to ensure a real-time implementation, as i_{max} depends on the choices made by the user on simulation time and integration step.

B. GRAMPC LIBRARY ORGANIZATION

The GRAMPC framework is designed to be portable and executable on different operating systems and hardware without the use of external libraries. The code is implemented in plain C with a user-friendly interface to C++, Matlab/Simulink, and Dspace. As schematized in Fig. 1, the main files are:

i **probfct.c** in which the structure of the dynamic equations, constraints and cost function are defined;
ii **grampc_RUN.c** in which the Augmented Lagrangian formulation and the OCP problem through the gradient-descent algorithm are implemented; this function needs to invoke **ffct** since is the one which define the model equations;
iii **main_SYSTEM.c** in which the other functions are invoked, computing optimal (or sub-optimal) solution and evaluate the state vector evolution under such control solution.

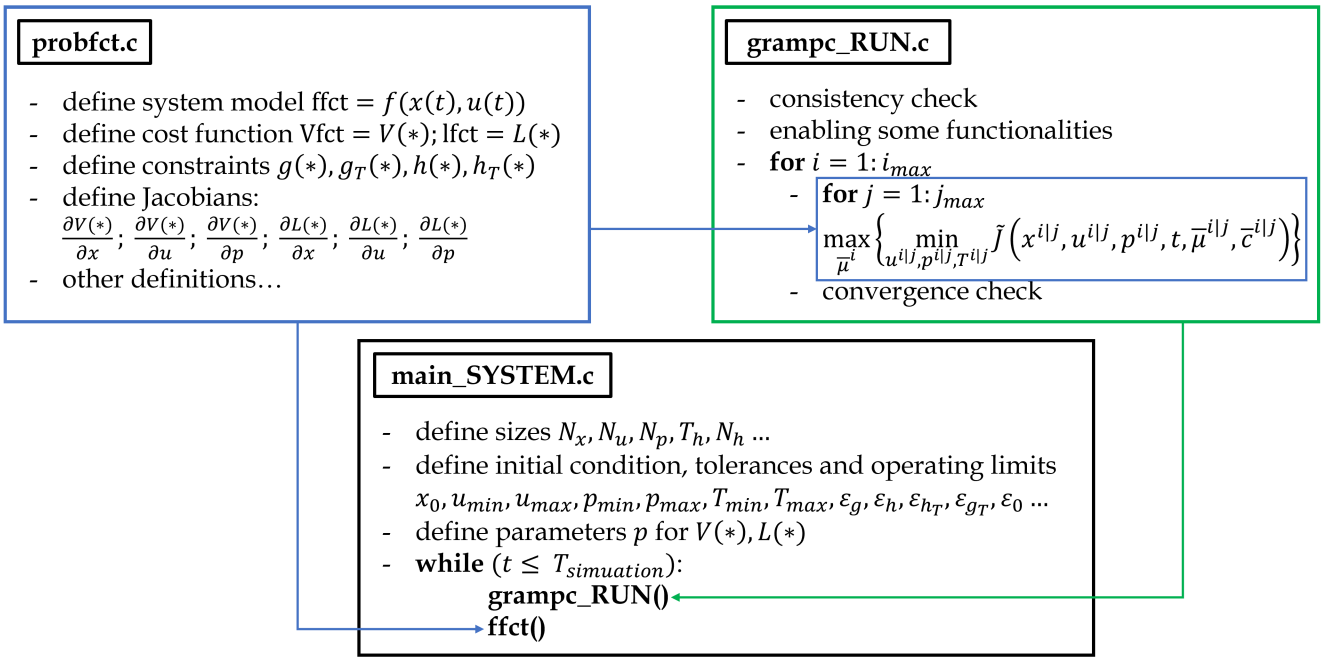


FIGURE 1. GRAMPC main files organization.

- Initialize $u^{i|1}, p^{i|1}$ and $T^{i|1}$
  Compute $x^{i|1}$ and set the scaling factor γ_p and γ_T
- ***for*** $j = 1: j_{max}$ ***do***
  - Compute $\lambda^{i|j}$ (by numerical integration of $\dot{\lambda}^{i|j}$)
  - Compute gradient components:
    $d_u^{i|j} = H_u\left(x^{i|j}, u^{i|j}, p^{i|j}, \lambda^{i|j}, t, \mu^i, c^i\right)$
    $d_p^{i|j} = \tilde{V}_p\left(x_{T^{i|j}}^{i|j}, p^{i|j}, T^{i|j}, \mu_T^{i|j}, c_T^{i|j}\right) + T^{i|j} H_p^{i|j}\left(T^{i|j}\right)$
    $d_T^{i|j} = \tilde{V}_T\left(x_{T^{i|j}}^{i|j}, p^{i|j}, T^{i|j}, \mu_T^{i|j}, c_T^{i|j}\right) + H^{i|j}\left(T^{i|j}\right)$
  - Update the Learning rate $\alpha^{i|j}$ by solving
    $\min_{\alpha>0} \tilde{J}(x^{i|j}, p^{i|j}, T^{i|j}, d_u^{i|j}, d_p^{i|j}, d_T^{i|j}, \alpha)$
  - Update u, p and T computing $u^{i|j+1}, p^{i|j+1}$ and $T^{i|j+1}$
  - Compute state vector $x^{i|j+1}$ by numerical integration of
    $f(x^{i|j+1}, u^{i|j+1}, p^{i|j+1}, t)$
  - Evaluate inner convergence criterion:
    $\eta^{i|j+1} = max\left\{\frac{\left\|u^{i|j+1} - u^{i|j}\right\|}{\left\|u^{i|j+1}\right\|}, \frac{\left\|p^{i|j+1} - p^{i|j}\right\|}{\left\|p^{i|j+1}\right\|}, \frac{\left|T^{i|j+1} - T^{i|j}\right|}{\left|T^{i|j+1}\right|}\right\}$
  - ***if*** $\left(\eta^{i|j+1} \leq \varepsilon_0\right)$ ***OR*** $j == j_{max}$ *then* ***break***;

end for

List 2. Pseudo routine of the inner cycle (min problem).

As schematized in Fig.2, the workspace of a GRAMPC project as well as algorithmic options and parameters are stored by the structure variable **grampc**.

Several parameter settings are problem-specific and need to be provided, whereas other values are set to default values.

A generic interface (Cmex interface) allows the communication among Matlab-files and C-code-files and allows to manipulate the grampc structure to set algorithmic options or parameters for the problem at hand.

The functionalities of GRAMPC can be manipulated from Matlab/Simulink by means of mex routines that are wrappers for the corresponding c-files.

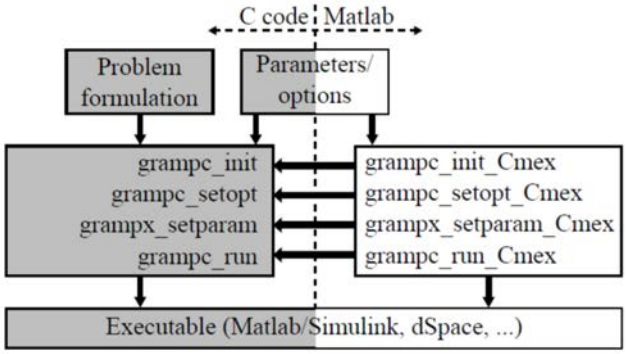


FIGURE 2. Interfacing of GRAMPC to C (grey) and matlab/simulink (white).

III. REFERENCE MODELS FOR VEHICLES

A. KINEMATICS OF A VEHICLE

The "Single-Trace model" or "2D bicycle model" can be expressed as a simplified car model, schematized in Fig.3.

This is a classic model that does very well at capturing vehicle motion in normal driving conditions [29].

The time-continuous model describing the kinematics of a vehicle is given Equations 11.

Where x and y are the coordinates of the center of mass in an inertial frame (X, Y). ψ is the inertial heading and v is the speed of the vehicle. L_f and L_r represent the distance from the center of the mass of the vehicle to the front and rear axles, respectively.

β is the angle of the current velocity of the center of mass with respect to the longitudinal axis of the car. a is the acceleration of the center of mass in the same direction as the velocity.

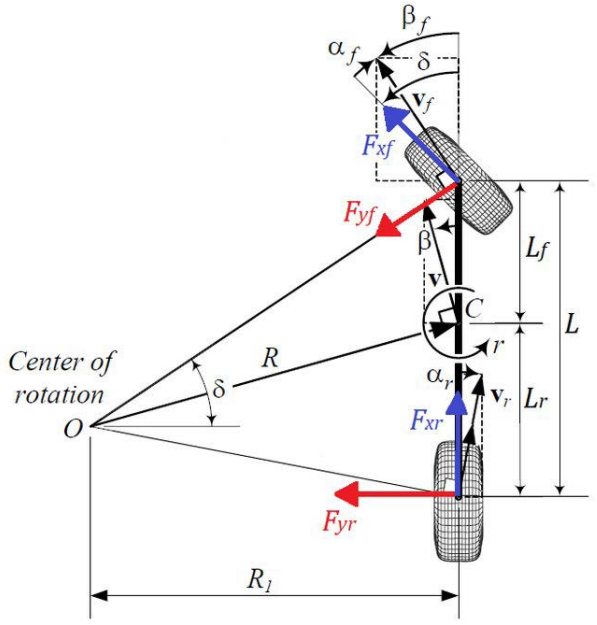


FIGURE 3. Single-trace model of a vehicle.

The control inputs are the front and rear steering angles δ, and a.

$$\begin{aligned}\dot{x} &= v\cos(\psi + \beta_f(\delta))\\ \dot{y} &= v\sin(\psi + \beta_f(\delta))\\ \dot{V} &= a\\ \dot{\psi} &= \frac{v}{L_\mathrm{r} + l_\mathrm{f}}\cos(\beta_f(\delta))\tan(\delta)\\ \beta_f(\delta) &= \arctan\left(\tan(\delta)\frac{L_\mathrm{r}}{L_r + L_\mathrm{f}}\right)\end{aligned} \tag{11}$$

B. DYNAMICS OF A VEHICLE

The kinematic model lacks references to forces opposing motion, such as the aerodynamic lift force, and reaction forces between the road surface and the tyres.

In the dynamic model it is possible to include this level of detail. The reference time-continuous model is given in the set of Equations (12).

In this case, the state vector is represented by $\begin{bmatrix}V_x & V_y & x & y & \psi\end{bmatrix}$ while the control vector is represented by $\begin{bmatrix}\delta & F_{xf}\end{bmatrix}$. F_{xf} is the traction force.

$$\begin{aligned}m(\dot{V}_x + V_y\omega) &= (F_{xf}cos(\delta)) - (F_{yf}sin(\delta)) - F_{air}\\ m(\dot{V}_y - V_x\omega) &= (F_{xf}sin(\delta)) - (F_{yf}cos(\delta)) + F_{yr}\\ I\ddot{\psi} &= L_f(F_{xf}sin(\delta) + F_{yf}cos(\delta)) - L_rF_{yr}\\ \dot{x} &= V_xcos(\psi) - V_ysin(\psi)\\ \dot{y} &= V_ycos(\psi) + V_xsin(\psi)\\ \dot{\psi} &= \omega\end{aligned} \tag{12}$$

where $F_{yf} = -C(\alpha, F)\alpha_f = -C(\alpha, F)[\frac{V_y + L_f\omega}{V_x} - \delta]$ is the lateral tire force on front tires with C_f cornering stiffness of front tires and α is the tire slip angle.

TABLE 1. Nomenclature of tested mathematical models.

Model	Set of Equations
$K4$	classic kinematics reported in (11)
$D6$	classic dynamics reported in (12)
$K6$	combination of (11) + (14)
$D8$	combination of (12) + (14)

$F_{yr} = -C(\alpha, R)\alpha_r = -C(\alpha, R)[\frac{V_y + L_r\omega}{V_x}]$ is the lateral tire force on rear tires with C_r cornering stiffness of rear tires and α_r is the tire slip angle.

$F_{air} = \frac{1}{2}\rho_{air}C_xSV_x^2$ is the frictional force of air, where ρ_{air} is the air density, C_x is the penetration coefficient depending on the shape of the vehicle and S is the frontal area of the vehicle. $C(\alpha, R)$ and $C_(\alpha, F)$ are called "stability derivatives", and although they depend on the tyre characteristics, values typically used in the literature can be assumed.

In fact, the important variations occur in the analysis of racing vehicles.

C. AUGMENTED MODELS FOR CONTROL

With the idea of improving performance, the dynamics of the vehicle's position and yaw error were introduced as auxiliary equations to those of the two models described above.

In particular, the problem of tracking the trajectory that describes the centre of a lane, on a road whose description in Cartesian coordinates is known. This makes it possible to write the following auxiliary equations.

$$\begin{aligned}e_x &= x^* - x\\ e_y &= y^* - y = f(x) - y\\ e_\psi &= \psi^* - \psi = \theta(x) - \psi\end{aligned} \tag{13}$$

In the following it is assumed that the reference for the x-coordinate of the vehicle is linearly dependent on the time variable.

This allows us to describe the road section as a function of the type $f(x) = f_0sin(x)$ and to describe the yaw angle of the vehicle in a simple way as $\theta(x) = \arctan(f(x))$.

$$\begin{aligned}\dot{e}_y &= \frac{\partial f(x)}{\partial x}\frac{\partial x}{\partial t} - \dot{y} = \frac{\partial f(x)}{\partial x}\dot{x} - \dot{y} = F(x)\dot{x} - \dot{y}\\ \dot{e}_\psi &= \frac{\partial \theta(x)}{\partial x}\frac{\partial x}{\partial t} - \dot{\psi} = \frac{\partial \theta(x)}{\partial x}\dot{x} - \dot{\psi} = \Theta(x)\dot{x} - \dot{\psi}\end{aligned} \tag{14}$$

The Equations (14) are used to increase the dynamics of the mathematical models presented above, to "force" the error for trajectory tracking to tend to a null value, exploiting the knowledge of path representation in Cartesian coordinates.

Since the expression of $\dot{x}$, $\dot{y}$ and $\dot{\psi}$ changes between kinematic and dynamic models, the auxiliary equations change their appearance in the various use cases.

From here on we refer to the following nomenclature, given in Table 1, concerning the models used in the tests.

IV. SOFTWARE-IN-THE-LOOP ANALYSIS

In the following, a verification of the functionality of the *GRAMPC* algorithm on two different control problems:

TABLE 2. Model parameters in SIL and PIL simulations.

Parameter	Value
δ_{min}	-1.2 rad
δ_{max}	+1.2 rad
a_{min}	- 11.20 $\frac{m}{s^2}$
a_{max}	+ 5.34 $\frac{m}{s^2}$
m	2200 kg
L_r	1.670 m
L_f	1.394 m
r_0	25.5 cm

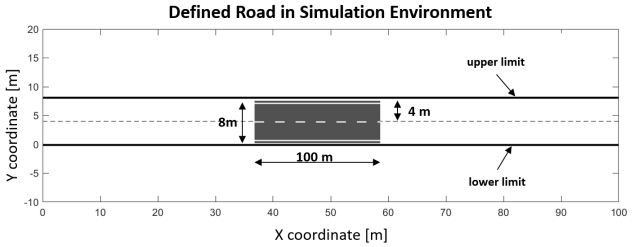


FIGURE 4. Straight road, without obstacles.

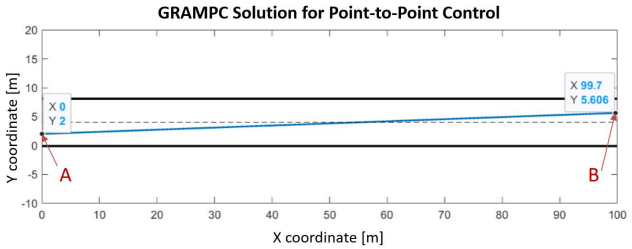


FIGURE 5. Controlled vehicle on straight road without obstacle.

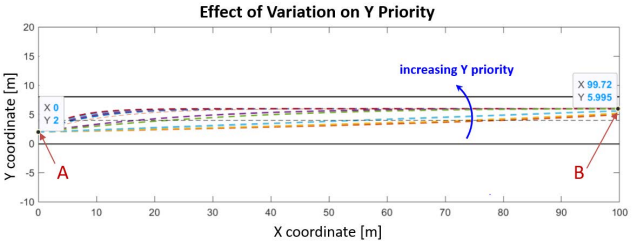


FIGURE 6. Effect of weight variation.

(a) movement of the vehicle from an a point A to a point B (seen as a way-point) both with a clear roadway and with obstacles; (b) trajectory tracking that describes the course of the carriageway (assuming it can be described implicitly in 2D space) both with a free carriageway and with obstacles.

With regard to the definition of obstacles, based on the dimensions of the vehicle, we consider a constraint area increased by an amount equal to half the longitudinal thickness of the vehicle, to take into account the fact that we control the coordinates of the vehicle's centre of gravity.

In Tab.2 are reported the parameters used both in SIL and PIL sections.

A. POINT-TO-POINT CONTROL

1) WITHOUT OBSTACLES

First of all it is defined a working environment on which to perform the relevant simulations. It has been supposed to work on a straight, unobstructed road.

The dimensions of the single carriageway are 4 m and a 100 m long road is assumed, as reported schematically in Fig. 4.

In the first simulation we are going to analyse the first important function of the MPC controller.

The system, defined through its model, generates a vector of state variables. This vector will contain all the state variables of the system that are linked to the system itself through physical equations of their evolving in time.

All the state variables that appear in the system can be controlled; all of them, in fact, known their initial value at time 0 (imposed in the project phase) can be controlled by the MPC system to reach a certain desired value at the end of the simulation.

In our case we will only deal with the variables that control the position on x and y in a top view of the system. All the auxiliary variables that control the system, if not directly controlled, adjust to meet the conditions imposed in the design phase.

We will therefore place a strong weight in the cost function on minimising the difference between the initial and desired X_{des} and Y_{des} positions.

The aim of this first part will therefore be to reach a desired point starting from an initial point. In particular $X_0 = [0\,2]^T, X_{des} = [100\,6]^T$. The system respects the conditions going from the initial point to the final point in a predetermined time, simulation time.

Fig. 5 shows the trajectory of the centre of mass of the vehicle when it is required to start from one point and arrive at another point, both known a priori, having fixed some values of the weights of the objective function.

It can be seen that having set the same "priority" on X and Y coordinates, the trajectory resulting from the resolution of the optimality problem is essentially a straight line. As can be seen, the variation of these parameters affects the trajectory as shown in Fig. 6.

2) WITH OBSTACLES

As anticipated, in case of obstacles each is interpreted as a forbidden area, described in terms of inequality constraints for the control problem, see Fig. 7.

Fig. 5 simulates a car travelling on a straight carriageway with starting position at $t = t_0$ and desired position at $t = t_{sim}$ and a possible obstacle in the middle of the carriageway. In the simulations shown below we set $t_{sim} = 10s$.

The most important parameter for the obstacle avoidance function is certainly the prediction horizon, since according to this parameter, the vehicle "sees" the road ahead closer or further away.

Fig. 8 shows the results with a "nominal" choice of the tuning parameters in the point-to-point control problem.

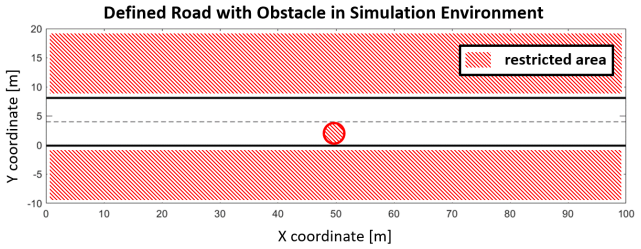


FIGURE 7. Road with constraints.

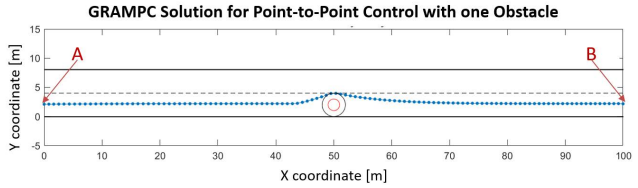


FIGURE 8. Straight road point to point with one obstacle.

Fig. 9 shows the variation of the trajectory according to the change of the prediction horizon. Another interesting analysis of the variation of the parameters of the control optimisation problem is that relating to the various combinations of priorities, as shown in Fig. 10.

A very interesting result is the one represented by the green line, in Fig. 10, related to a configuration with high priority to the constant velocity constraint.

The solution that the algorithm finds is to ''go around'' the endpoint. Intuitively, this is a solution that makes ''physical'' sense, since high priority is given to maintaining constant velocity, the solution that is ''closest'' to the goal of reaching the desired end position is certainly to go around it.

Similar results are obtained by inserting more than one obstacle constraint during the route. For example, Fig. 11 shows the case in which the car shall maintain the centre-line of the roadway (imposing the same height at the point of departure and arrival) and there are four obstacles placed alternatively in the two lanes.

The results shown relate to an analysis of the ''feasibility'' of the solutions proposed by the GRAMPC algorithm when the kinematic vehicle model (K4) is taken into account.

Fig. 13 shows the comparison between the K4 and D6 models in the problem of avoiding the four obstacles and maintaining the centre-line of the roadway.

It can be seen that the differences in ''performance'' are hardly appreciable in terms of the trajectory taken by the vehicle.

In practice, the two trajectories overlap. However, Fig. 12 shows a difference in the solutions derived from the GRAMPC algorithm, highlighting that the control actions are more ''aggressive'' in the case of the kinematic model.

In Figures 12 and 14 each physical quantity has its own units: X, Y in metres; V_x, V_y metres/second; θ, ψ radians; ω in radians/second.

On the other hand, the use of a more complex model requires higher computation times, as shown in Fig. 15.

Note however that, having fixed the integration time at 10 *ms*, both are characterised by relatively low computational complexity. In fact, as shown in Fig. 16, the time required to simulate a 10 ms model step and then derive the MPC solution remains smaller (on average) at 0.5 ms for both vehicle models considered.

This kind of consideration is useful to state that the algorithm can be considered to be implemented in real-time.

In Figure 15 is shown the computation time in milliseconds, with respect of simulation time in seconds, both for kinematic and dynamic models.

B. TRAJECTORY TRACKING CONTROL

The second control problem addressed is that of trajectory tracking in the XY plane, which explicitly describes the ''shape'' of the roadway.

It is assumed that the X coordinate of the vehicle evolves linearly in time and that the desired Y coordinate is a sinusoidal function of X itself.

Similarly, as regards the yaw angle of the vehicle, it is assumed that the centre line (in the forward direction) of the vehicle must be tangent to the trajectory at all times.

The results proposed in Fig. 17 are related to the best combination of priority coefficients and prediction horizon derived from the preliminary analysis in the driving problem with way-points, so as to make an equal coefficient analysis in the optimization problem setting.

The figure shows that in the case of the K4 and D6 models there is a violation of the obstacle constraint, as the trajectory of the vehicle's centre of mass enters the ''safety'' circle.

The same is true for the K6 model while the D8 model fulfils both objectives, trajectory tracking at the same time as obstacle avoidance.

Certainly D8 is the most complicated model, in fact as it can be seen from Fig. 16 the computation times associated with D8 are those with the highest peak values.

However, to perform the 10 ms simulation of the evolution of the dynamic model, the processor always takes less than 2 ms. This means that D8 is also a good candidate for implementation on an embedded platform.

In Fig. 20(a) a robustness analysis to the variation of the initial condition of the vehicle yaw angle is presented.

In particular, the case in which the track constraints are set as ''soft'' constraints that do not affect the solution obtained by the GRAMPC algorithm very much is shown.

Intuitively, different starting angles mean that the transitional phase evolves differently.

Fig. 20(b) shows the parametric robustness analysis, combining variations in vehicle attitude angle with variations in initial vehicle position.

It can be seen that having set the constraints on the roadside, as ''soft'' the solutions computed by GRAMPC can be outside the roadway.

This is due to the fact that maintaining a constant vehicle speed is also one of the constraints on the optimisation problem.

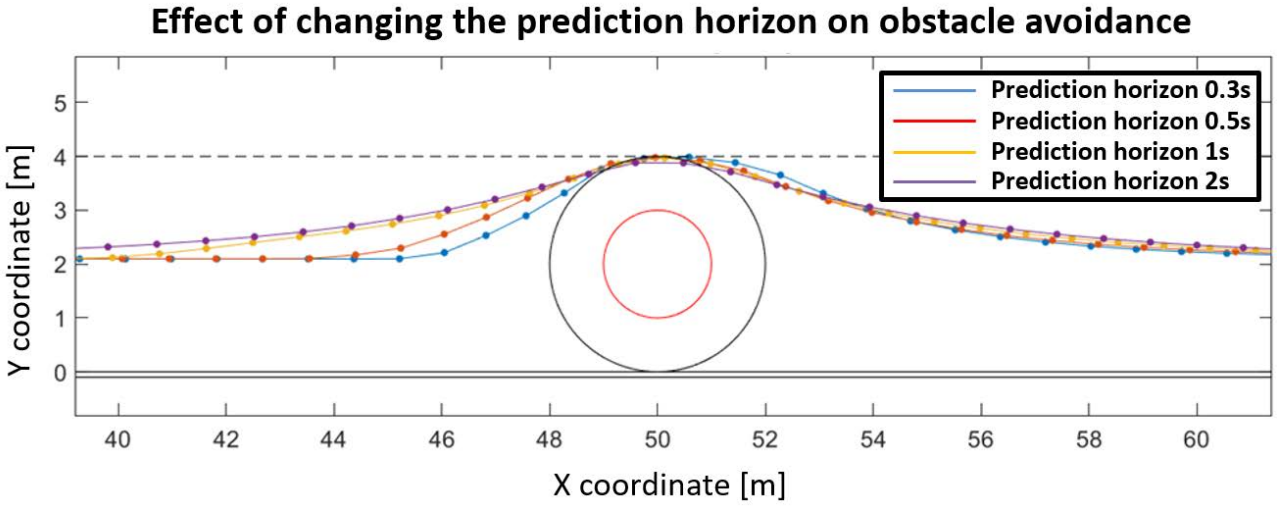


FIGURE 9. Overcoming an obstacle with a variable prediction horizon.

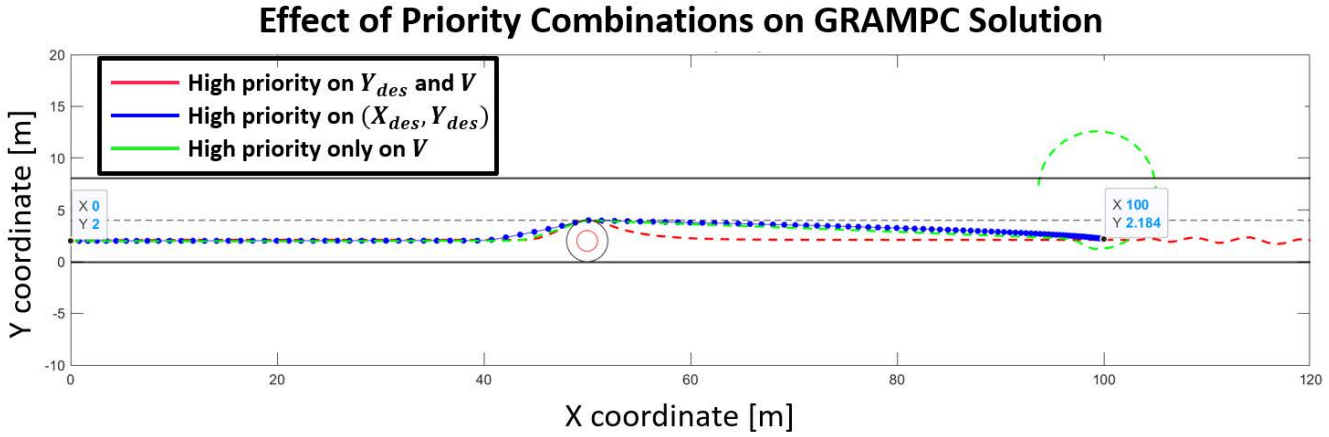


FIGURE 10. Combination of the priorities.

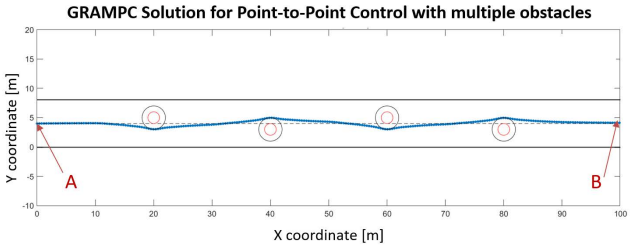


FIGURE 11. Controlled trajectory from point A to point B with obstacle.

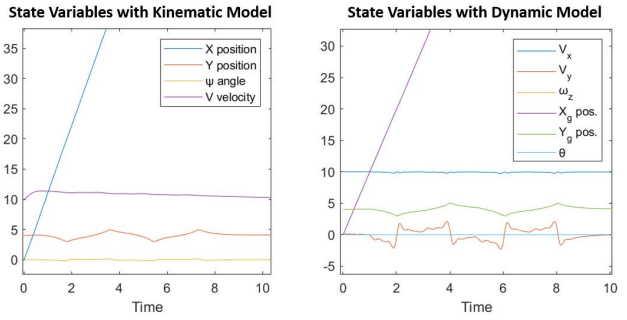


FIGURE 12. K4 VS D6 in terms of state variable evolution.

V. PROCESSOR-IN-THE-LOOP ANALYSIS AND VALIDATION

In this section the analysis of the GRAMPC algorithm is presented again, when the project is compiled on an Embedded platform. In particular, the Raspberry Pi 3 Model B with 64-bit 1.5 GHz quad-core Arm Cortex-A53 CPU is used, which is representative of the calculation capabilities of an automotive gateway.

The differences between the vehicle trajectories obtained on a i7 CPU in Section III and those obtained here on the embedded system are not appreciable. The substantial differences are in the calculation times.

An exhaustive analysis of the variation of the average computation time is proposed below, both with regard to the

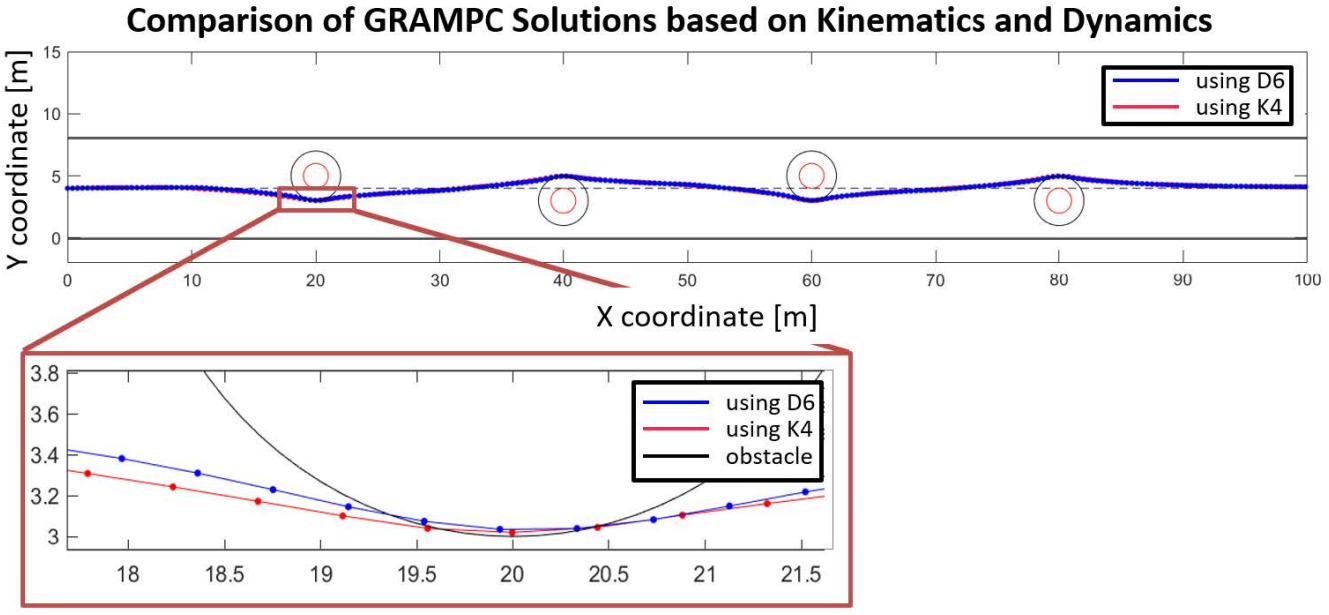


FIGURE 13. Comparison between K4 and D6 models.

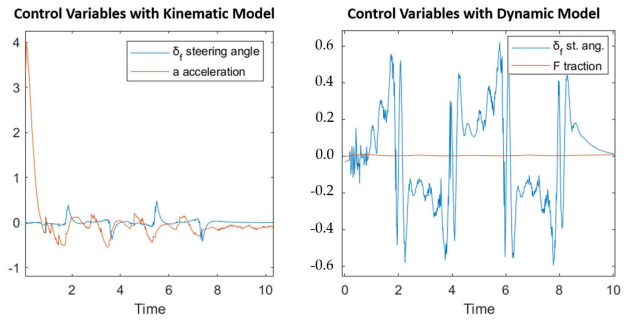


FIGURE 14. K4 VS D6 in terms of control variable evolution.

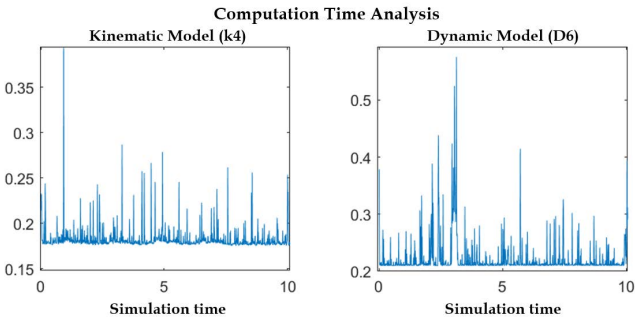


FIGURE 15. K4 VS D6 in terms of computation cost analysis.

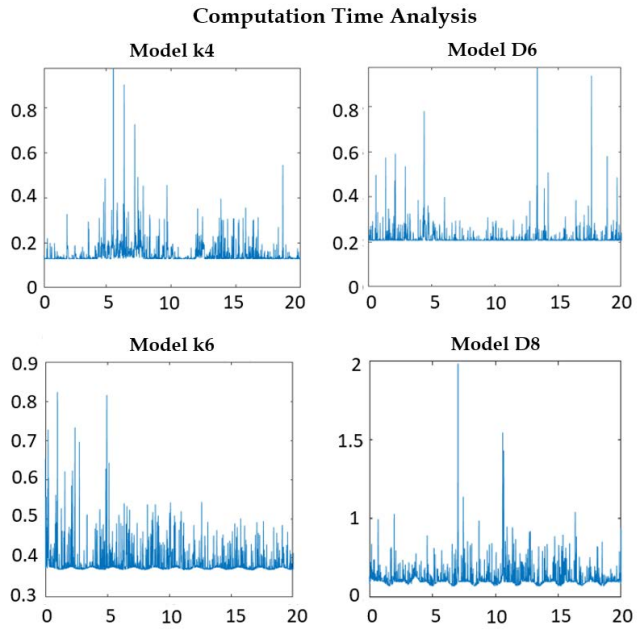


FIGURE 16. Computational analysis of the four different exploited models.

models used and to the numerical approximation method. The results on the embedded platform are obtained running the GRAMPC algorithm as the only process task.

A. EFFECT OF MODEL CHOICE

Computation time analysis is proposed by comparing K4 and D6 models on point-to-point displacement problems with both one and four obstacles placed on the roadway, as shown in the MIL simulation section.

In addition, the K6 and D8 models are compared, augmented by the error dynamics equations in the case of 2D trajectory knowledge (Equations (14)).

Fig. 18 shows the variation of the average computation time with respect to the choice of model on which GRAMPC bases the solution of the optimisation problem.

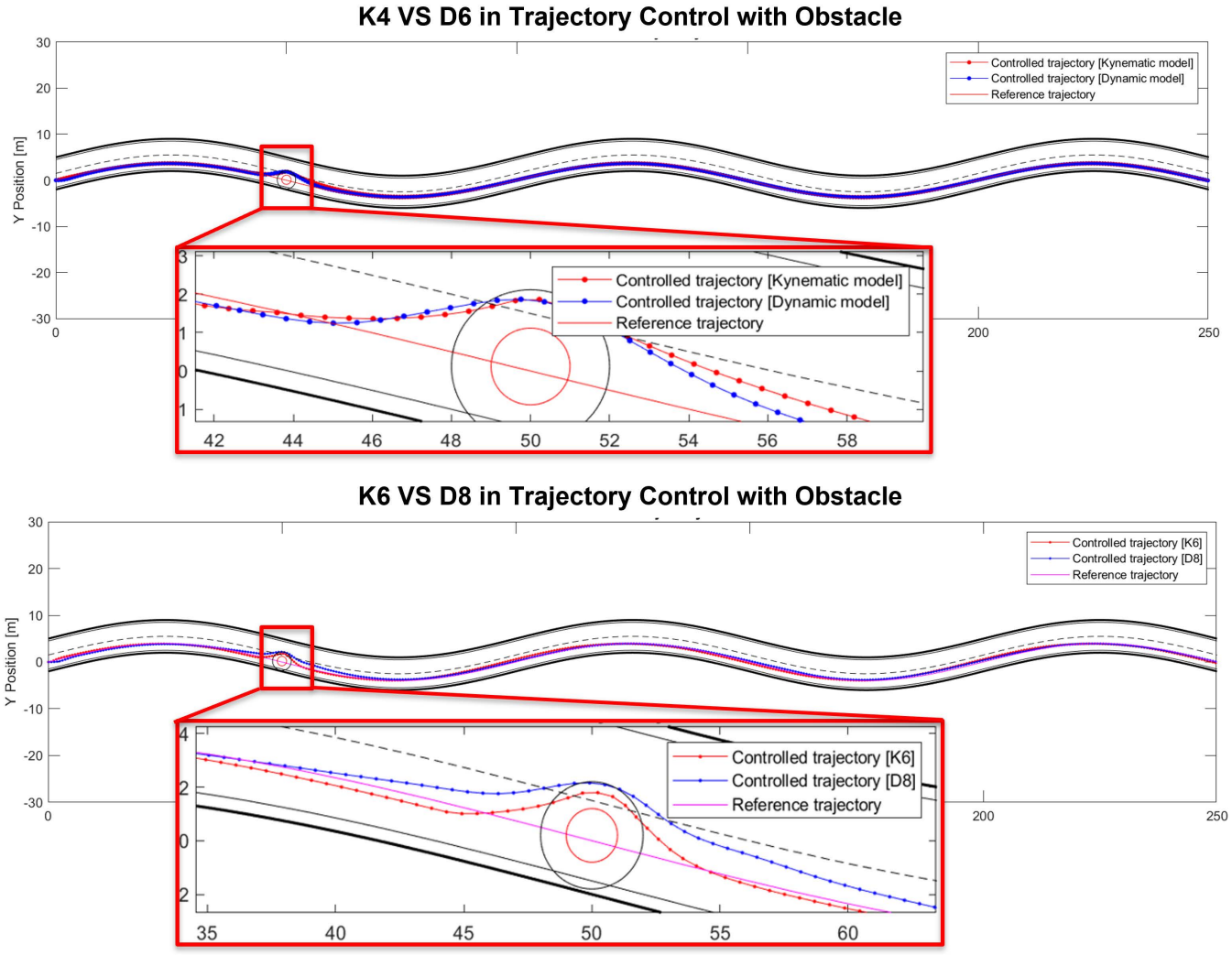


FIGURE 17. Comparison between "classic" models (K4 and D6) and "augmented" models (K6 and D8).

In both cases, with one and four obstacles respectively, D6 is computationally more expensive. Note that the initial computation time exceeds the real time limits.

This fact is clearly due to the initial stages of compiling and starting the library in wrapped form and the simulation environment.

However, it should be noted that the average computation times are well below the chosen integration time of 10 ms (indicative for "real-time" applications).

Similar comparison, on the trajectory tracking problem with an obstacle on a roadway, is made with respect to the augmented models, K6 and D8 respectively, as shown in Fig. 19. Table 3 summarises the results in Figs. 18 and 19, in terms of the average computational time recorded on the simulations carried out, varying the vehicle models.

It should be noted that the various tests were carried out by setting the numerical solver as the one based on Euler's numerical method. As we expected, the D8 model is certainly the most complex to process for the GRAMPC numerical optimiser. In any case, it is also pointed out that the variation of the model with which the vehicle is described has a rather manageable influence in terms of processing times.

TABLE 3. Summary of the computational-time analysis.

Used Model	Computational-time [ms]
K4	0.653 (with 1 obstacle)
	0.658 (with 4 obstacles)
D6	0.726 (with 1 obstacle)
	0.782 (with 4 obstacles)
K6	1.226 (trajectory tracking)
D8	2.573 (trajectory tracking)

B. EFFECT OF NUMERIC METHOD CHOICE

The GRAMPC library provides four different numerical methods for approximating differential equations.

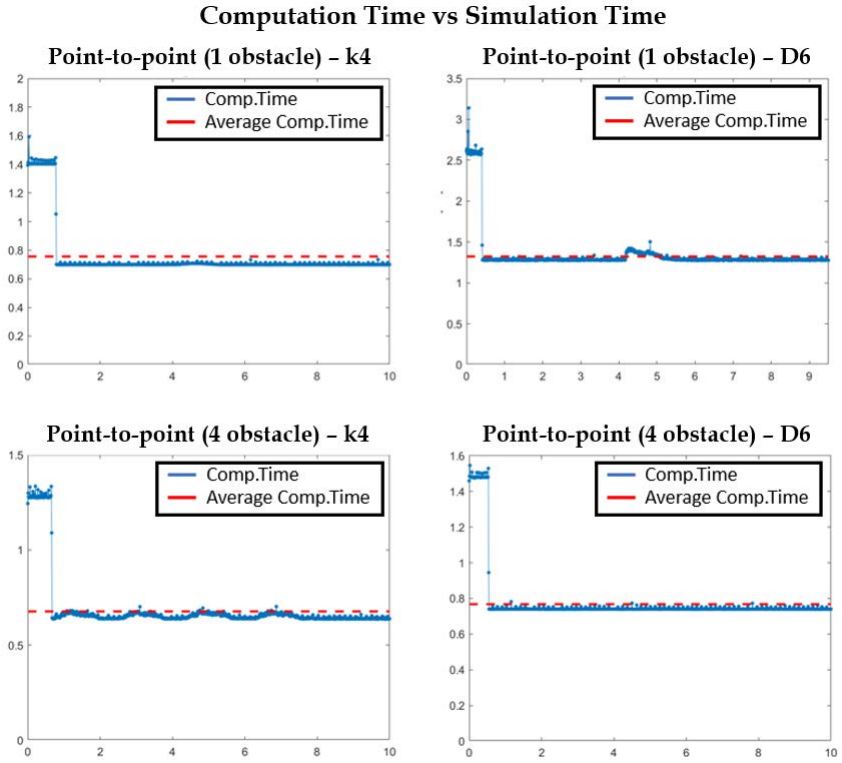


FIGURE 18. Effect of variation of the model for "Point-to-Point" control problems (Comp.Time in blue and Average Comp.Time in red).

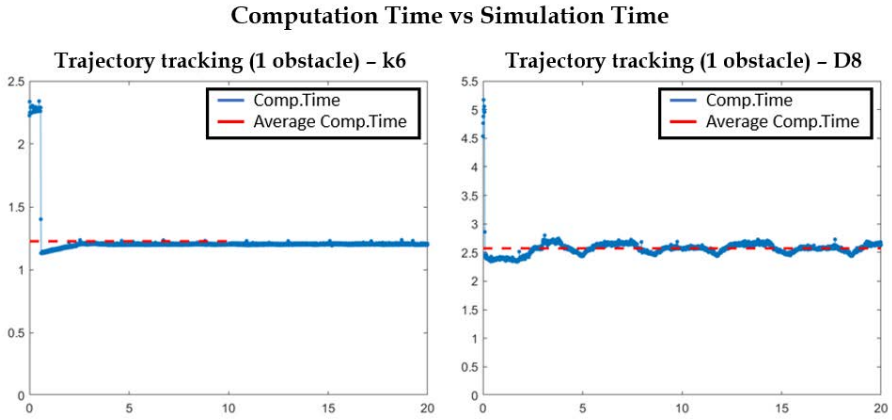


FIGURE 19. Effect of variation of the model for "Trajectory Tracking" control problems (Comp.Time in blue and Average Comp.Time in red).

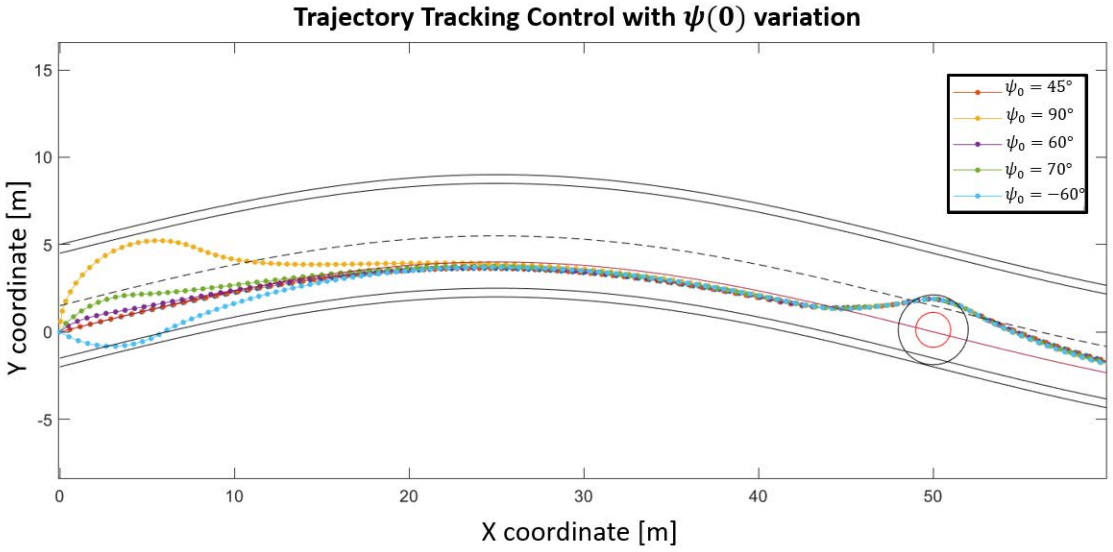
In particular in this work we compares the Euler, Heun, Runge-Kutta and Rosenbrock methods [30].

Fig. 21 shows a comparison between the trajectory obtained by GRAMPC when setting the Euler method (orange trajectory) and the Rosenbrock method using the RODAS solver (blue trajectory), respectively.

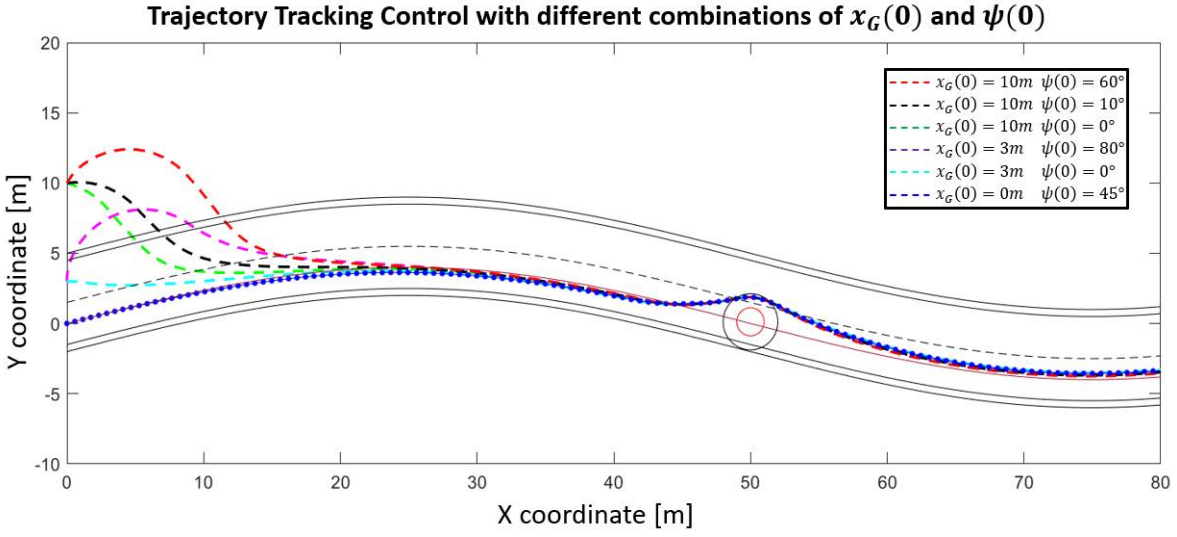
Fig. 23 shows the comparison between the tracking error of the trajectory describing the roadway, in particular on the Y-coordinate. It is shown that the use of Euler's method improves performance compared to the use of a computationally heavy method such as Rosenbrock one.

This is justified graphically through the analysis of computational times in Fig. 22 that shows the differences in terms of computational time required with the different numerical methods provided by GRAMPC.

As far as the implementation on Raspberry Pi 3 B+ is concerned, the methods that are suitable for maintaining a "real-time" throughput are Euler and Heun, while the computation



(a)



(b)

FIGURE 20. (a) Effect of $\psi(0)$ variation; (b) Combined effect of $x_G(0)$ and $\psi(0)$ variations.

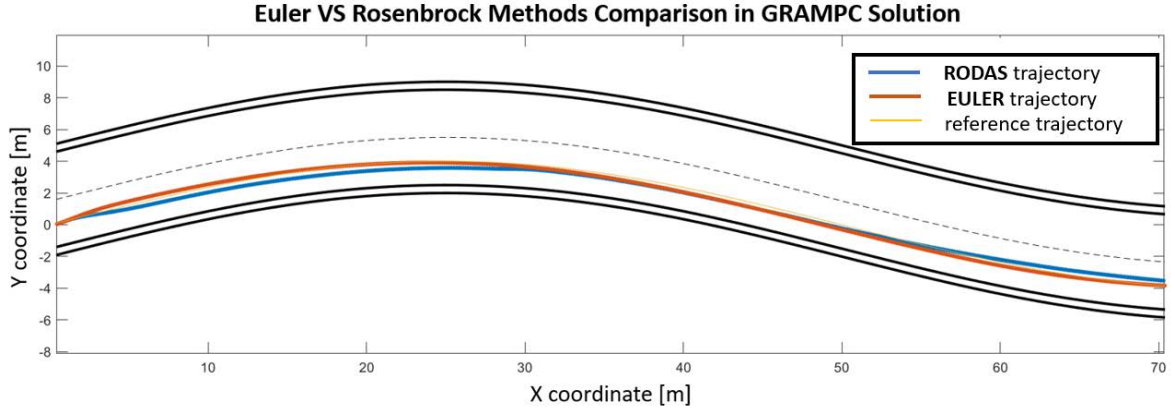


FIGURE 21. Precision comparison with different numerical approximations.

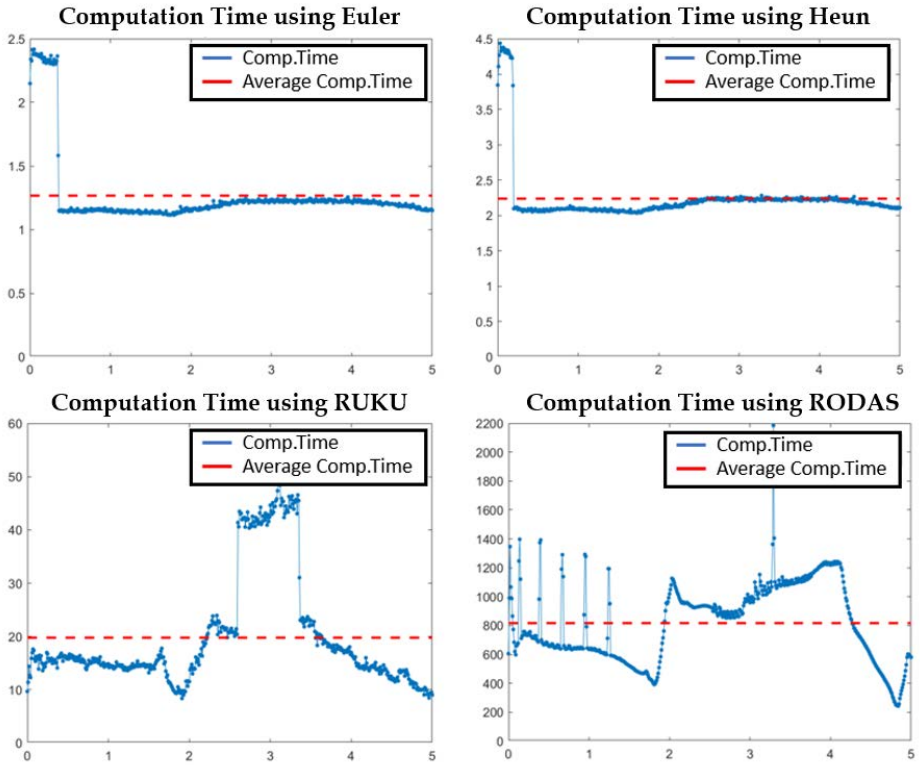


FIGURE 22. Comparison of required computational time with variation of the numerical methods (Comp.Time in blue and average comp.time in red).

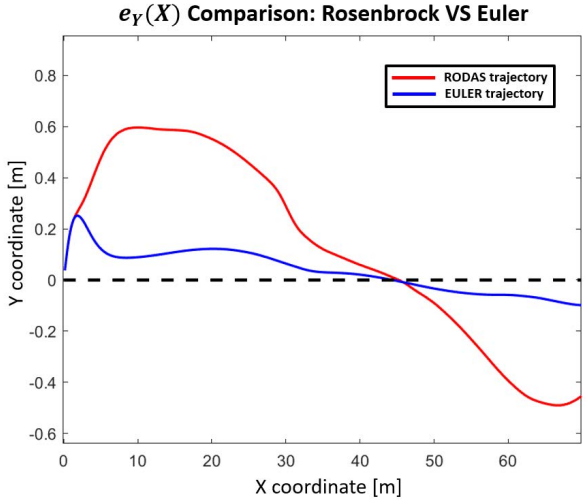


FIGURE 23. Trajectory tracking error e_Y with different numerical methods.

times with the Rosenbrock (RODAS) and Runge-Kutta (RUKU) methods are too high compared to the reference sampling time of 10ms.

Table 4 summarises the results of the variation analysis of the numerical approximation method, shown in Fig. 22.

C. REAL-TIME ROBUST MPC WITH MEASUREMENT NOISE INJECTION

A fundamental question about an MPC system is its robustness to model uncertainty and noise from external sources.

TABLE 4. Summary of the computational-time analysis.

Numerical Method	Avg. Computational-time [ms]
Euler	1.267
Heun	2.238
Runge-Kutta (RUKU)	19.911
Rosenbrock (RODAS)	809.234

From the control theory, a control system is robust when the stability is maintained and the performance specifications are met for a specified range of model variations and a class of noise signals (uncertainty range).

To be meaningful, any statement about ''robustness'' of a particular control algorithm must make reference to a specific uncertainty range. In this context, an Additive White Gaussian Noise generator [31] has been developed to introduce random errors into the states vector.

In particular, a single AWGN generator has been exploited for X and Y positions, yaw angle and velocity to insert possible external disturbances, coming from a real environment. The standard deviation chosen for these generator is characteristic for every state variable and has been sized considering a possible real error introduced in the system.

Two types of AWGN generator have been implemented for the MPC algorithm developed: one with a standard deviation of 0.065, that produces a maximum absolute error of 0.25 and it's added to X and Y position and to the velocity. The other generator has a standard deviation of 0.002, with a maximum

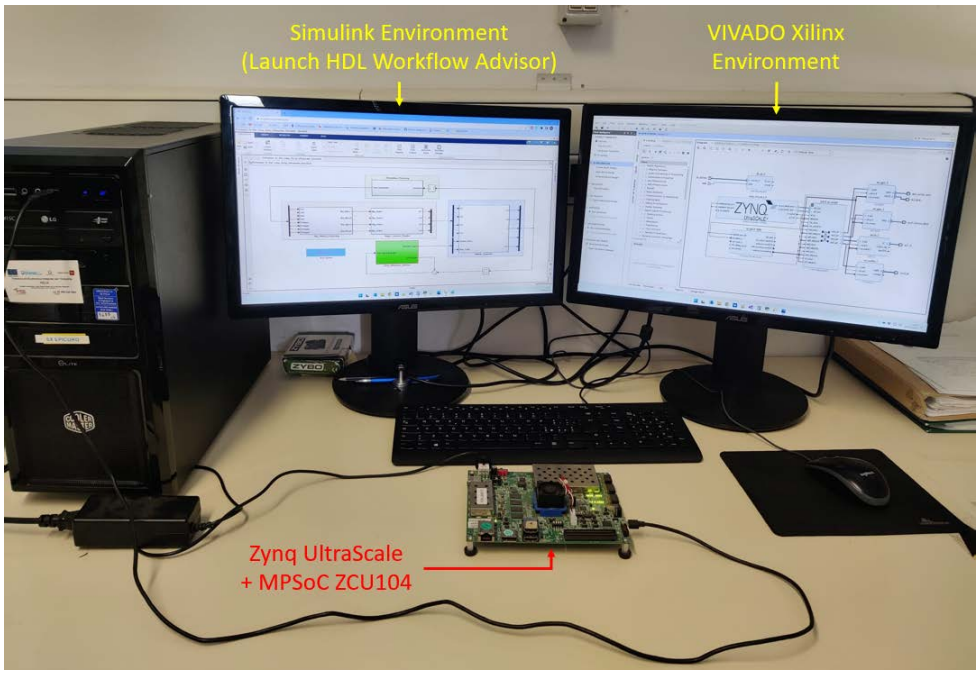


FIGURE 24. Setup configuration for processor-in-the-loop validation with zynq ultrascale + MPSoC ZCU104.

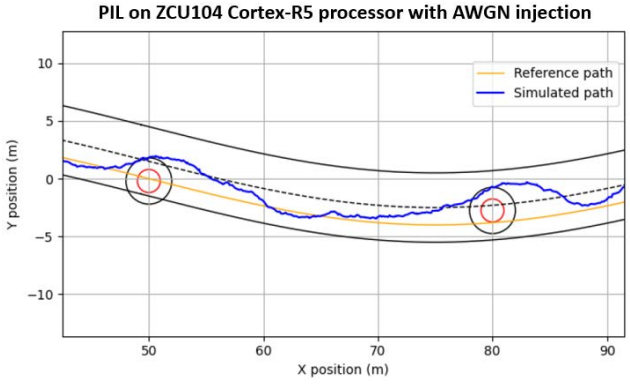


FIGURE 25. Effect of the noise injection in the control loop.

absolute error of 0.0075 and it's used to characterize the yaw angle state.

With this solution, the future iteration receives as initial state vector a set of variables that is different from the optimal value calculated with the Optimal Control Problem developed. This forces the algorithm to a fast adaptation to restore the optimal state values to follow the reference trajectory and avoid the obstacles.

Analysing this aspect through the vehicle behaviour, it's possible to understand if the MPC algorithm has the robustness to correct possible random errors and to maintain the right trajectory.

Fig. 25 shows the solution of the GRAMPC algorithm when the feedback is perturbed by additive disruption, generated through the FPGA logic of ZynQ ZU7EV device, and the MPC algorithm runs on the programmable core available on the same device.

The ZynQ ZU7EV, mounted on the prototyping ZCU104 board in Fig. 24, has 2 types of programmable cores: a quad-core Cortex-A53 and a dual-core Cortex-R5. The simulation with disturbance injection was performed on both Cortex-A53 and Cortex-R5 processors.

The differences in the solution found by GRAMPC are not appreciable, so the results shown are relative to the use of the Cortex-R5, being a safe processor characterized by a lower power consumption than the Cortex-A53.

The implementation of AWGN generation and disturbance insertion on the ZU7EV device required 11700 LUTs and 160 DSP blocks, corresponding to 5% and 9%, respectively, of the available FPGA resources.

The simulations with disturbance injection are performed with the same control problem setting parameters in the nominal case. The performance could be improved by finding the most suitable combination to compensate for the presence of disturbances.

The objective in this case was to verify the robustness of the nominal solution in the case of additive disturbances in the feedback. For simplicity of exposition, the PIL simulation in the case of using the D8 model with Euler numerical approximation algorithm is shown.

D. ROBUSTNESS TO MODEL UNCERTAINTY

A further verification consists in testing the GRAMPC solution, inserting in the dynamic equations of the vehicle the tire-road surface interaction model of Pacejka [32], [33].

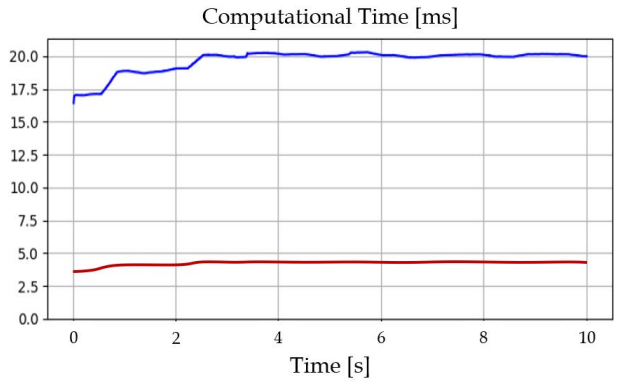


FIGURE 26. Computational time analysis with Pacejka model inserted only in Simulink Model (red) and also in GRAMPC runned on ZCU104 (blue).

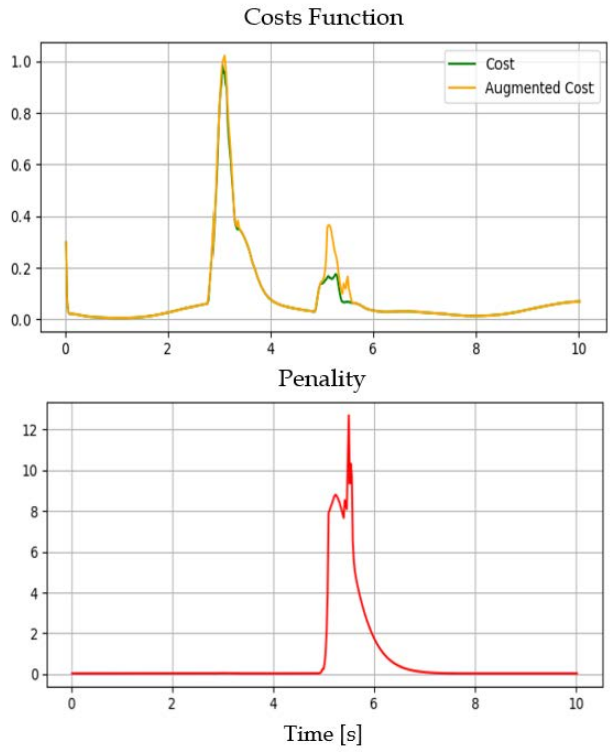


FIGURE 27. Comparison in cost function of original OCP VS augmented lagrangian formulation (up) and behaviour of Penality in case of Pacejka model only in vehicle simulator.

Fairly comparable results were obtained, both in the case where Pacejka is only in Simulink and GRAMPC uses the linear model, and in the case where Pacejka is also within GRAMPC.

As shown in Fig. 26, the more evident difference is that the computational time increases significantly even in the case of using numerical approximation with Euler method.

Indeed, in the case where Pacejka model is both in Simulink and on an Embedded platform, the computational time seems not to respect the limit for the real-time implementation, which in the specific case is 10ms.

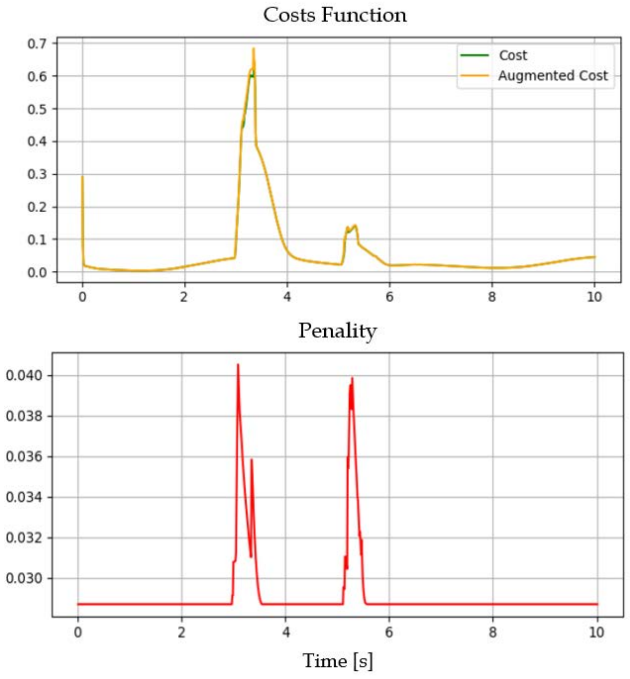


FIGURE 28. Comparison in cost function of original OCP VS augmented lagrangian formulation (up) and behaviour of Penality in case of Pacejka model is both in Simulink and GRAMPC on ZCU104.

In the case where Pacejka is used in Simulink and the linear model is used in GRAMPC on an Embedded system, as shown in Fig.27, the original cost functional and that of the Lagrangian function differ more markedly than in the case where Pacejka is used in both Simulink and ZCU104, reported in Fig.28.

GRAMC compensates for the difference between the system model and the internal model (the one used for prediction) by increasing the penalty coefficient.

In the case where Pacejka is both in Simulink and on ZCU104, being the system model and the prediction model, GRAMPC does not have to compensate for this deviation through a penalty, in fact the original cost functions and the Lagrangian formulation are substantially overlapping.

Reasonably, the manoeuvres required in the analysis proposed in this paper, do not require values of δ, α_f and α_r that highlight the differences between the two models used. Most likely, analyzing assisted parking maneuvers or trajectory requests with very tight curve sections would highlight the need to use a more accurate model even on the control algorithm side of the embedded system.

VI. CONCLUSION AND FUTURE WORK

This paper presented the design and real-time implementation on embedded systems of a novel MPC, incorporating obstacle avoidance functionalities and based on the GRAMPC library in which gradient descent is used as a numerical optimisation method.

The structure of the library and the operating procedure for using it and testing its functionality through the case study of vehicle control was presented.

In this work, an exhaustive analysis of the variations obtained on the solution calculated on the basis of the GRAMPC prediction has been presented, with regard to the parameters of the optimisation problem, the choice of the mathematical model describing the vehicle, and the numerical approximation methods.

The proposed analysis aims to find the trade-off between accuracy (i.e. the complexity of the mathematical abstraction required to describe the physical process) and computational complexity (i.e. the computational time required to calculate the prediction itself).

Modified equations have also been proposed for model augmentation, in order to solve a trajectory control problem, under the reasonable assumption that it is possible to mathematically describe the road route, at least around the vehicle's current position.

The analysis of the results shows that the choice of the model, although important, has less impact than the choice of the numerical method of approximation of the differential equations.

In particular, this gives important indications on the customised implementation of GRAMPC, without having to fully import the functions of all numerical methods, or wanting to directly implement low-level code.

Robustness analysis to measurement disturbances was presented, exploiting the potential of the ZCU104 platform, on which disturbance injection via HW acceleration was implemented, while the MPC algorithm was ported on a safe Cortex-R5 core with an HW/SW architecture that is fully compliant with the real-time and low-complexity/low-power requirements of automotive ECUs.

The achieved results well compare to the state of the art where advanced MPC algorithms are usually not implemented in real-time and/or are not suitable for resource constrained embedded systems.

Extensions of our work will include further analysis of the algorithm by exploiting advanced Co-simulation and Formal Verification paradigms [34].

Increasing the level of details in the mathematical model, including also power drive system [35] and power electronics, for the evaluation of particular properties, related to the behaviour of the vehicle in the face of the introduction of possible faults.

REFERENCES

[1] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car—Part I: Distributed system architecture and development process," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 7131–7140, Dec. 2014.

[2] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[3] A. Singandhupe and H. M. La, "A review of SLAM techniques and security in autonomous driving," in *Proc. 3rd IEEE Int. Conf. Robotic Comput. (IRC)*, Feb. 2019, pp. 602–607.

[4] K. Na, J. Byun, M. Roh, and B. Seo, "RoadPlot-DATMO: Moving object tracking and track fusion system using multiple sensors," in *Proc. Int. Conf. Connected Vehicles Expo (ICCVE)*, Oct. 2015, pp. 142–143.

[5] Á. Llamazares, E. J. Molinos, and M. Ocaña, "Detection and tracking of moving obstacles (DATMO): A review," *Robotica*, vol. 38, no. 5, pp. 761–774, May 2020.

[6] K.-W. Chiang, G.-J. Tsai, Y.-H. Li, Y. Li, and N. El-Sheimy, "Navigation engine design for automated driving using INS/GNSS/3D LiDAR-SLAM and integrity assessment," *Remote Sens.*, vol. 12, no. 10, p. 1564, May 2020.

[7] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, p. 2140, Mar. 2021.

[8] S. J. Anderson, S. C. Peters, T. E. Pilutti, and K. Iagnemma, "An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios," *Int. J. Vehicle Auto. Syst.*, vol. 8, nos. 2–4, pp. 190–216, 2010.

[9] G. Cesari, G. Schildbach, A. Carvalho, and F. Borrelli, "Scenario model predictive control for lane change assistance and autonomous driving on highways," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 3, pp. 23–35, Jul. 2017.

[10] E. Snapper. (2018). *Model-Based Path Planning and Control for Autonomous Vehicles Using Artificial Potential Fields*. Accessed: Mar. 2022. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid:453a26ea-8556-4927-8d9f-2058f7dcda15?collection=education

[11] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative UAVs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, 2017.

[12] S.-O. Park, M. C. Lee, and J. Kim, "Trajectory planning with collision avoidance for redundant robots using Jacobian and artificial potential field-based real-time inverse kinematics," *Int. J. Control, Autom. Syst.*, vol. 18, no. 8, pp. 2095–2107, Aug. 2020.

[13] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *Int. J. Control*, vol. 93, no. 1, pp. 62–80, 2016.

[14] J. Kalmari, J. Backman, and A. Visala, "A toolkit for nonlinear model predictive control using gradient projection and code generation," *Control Eng. Pract.*, vol. 39, pp. 56–66, Jun. 2015.

[15] F. Yakub and Y. Mori, "Comparative study of autonomous path-following vehicle control via model predictive control and linear quadratic control," *Proc. Inst. Mech. Eng., D, J. Automobile Eng.*, vol. 229, no. 12, pp. 1695–1714, Oct. 2015.

[16] F. Yakub, A. Abu, S. Sarip, and Y. Mori, "Study of model predictive control for path-following autonomous ground vehicle control under crosswind effect," *J. Control Sci. Eng.*, vol. 2016, Apr. 2016, Art. no. 6752671.

[17] H. Nam, W. Choi, and C. Ahn, "Model predictive control for evasive steering of an autonomous vehicle," *Int. J. Automot. Technol.*, vol. 20, no. 5, pp. 1033–1042, Oct. 2019.

[18] R. C. Rafaila and G. Livint, "Nonlinear model predictive control of autonomous vehicle steering," in *Proc. 19th Int. Conf. Syst. Theory, Control Comput. (ICSTCC)*, Oct. 2015, pp. 466–471.

[19] B. Weng, S. J. Rao, E. Deosthale, S. Schnelle, and F. Barickman, "Model predictive instantaneous safety metric for evaluation of automated driving systems," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 1899–1906.

[20] M. Ataei, A. Khajepour, and S. Jeon, "Model predictive control for integrated lateral stability, traction/braking control, and rollover prevention of electric vehicles," *Vehicle Syst. Dyn.*, vol. 58, no. 1, pp. 49–73, Jan. 2020.

[21] K. Han, G. Park, G. S. Sankar, K. Nam, and S. B. Choi, "Model predictive control framework for improving vehicle cornering performance using handling characteristics," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 3014–3024, May 2021.

[22] E. Enders, G. Burkhard, and N. Munzinger, "Analysis of the influence of suspension actuator limitations on ride comfort in passenger cars using model predictive control," *Actuators*, vol. 9, no. 3, p. 77, Aug. 2020.

[23] N. Guo, X. Zhang, Y. Zou, B. Lenzo, and T. Zhang, "A computationally efficient path-following control strategy of autonomous electric vehicles with yaw motion stabilization," *IEEE Trans. Transp. Electrific.*, vol. 6, no. 2, pp. 728–739, Jun. 2020.

[24] F. Cosimi, P. Dini, S. Giannetti, M. Petrelli, and S. Saponara, "Analysis and design of a non-linear MPC algorithm for vehicle trajectory tracking and obstacle avoidance," in *Proc. Int. Conf. Appl. Electron. Pervading Ind., Environ. Soc.* Cham, Switzerland: Springer, 2020, pp. 229–234.
[25] P. Dini and S. Saponara, "Design of adaptive controller exploiting learning concepts applied to a BLDC-based drive system," *Energies*, vol. 13, no. 10, p. 2512, May 2020.
[26] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 3, pp. 269–296, May 2020.
[27] Z.-P. Jiang, T. Bian, and W. Gao, "Learning-based control: A tutorial and some recent results," *Found. Trends Syst. Control*, vol. 8, no. 3, pp. 176–284, 2020.
[28] M. Cannon, W. Liao, and B. Kouvaritakis, "Efficient MPC optimization using Pontryagin's minimum principle," *Int. J. Robust Nonlinear Control: IFAC-Affiliated J.*, vol. 18, no. 8, pp. 831–844, 2008.
[29] M. Guiggiani, *The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars*, New York, NY, USA: Springer, 2014.
[30] A. Borzì, *Modelling With Ordinary Differential Equations: A Comprehensive Approach*. Boca Raton, FL, USA: CRC Press, 2020.
[31] X. Tan, Y. Liu, C. Zuo, and M. Zhang, "A real-time video denoising algorithm with FPGA implementation for Poisson–Gaussian noise," *J. Real-Time Image Process.*, vol. 13, no. 2, pp. 327–343, 2017.
[32] P. Shakouri, D. S. Laila, A. Ordys, and M. Askari, "Longitudinal vehicle dynamics using simulink/MATLAB," in *Proc. UKACC Int. Conf. Control*, 2010, pp. 955–960.
[33] MATHWORKS. *MATLAB/Simulink Documentation on Tire-Road Dynamics Given by Magic Formula Coefficients*. Accessed: Mar. 2022. [Online]. Available: https://it.mathworks.com/help/physmod/sdl/ref/tireroadinteractionmagicformula.html
[34] C. Bernardeschi, P. Dini, A. Domenici, M. Palmieri, and S. Saponara, "Formal verification and co-simulation in the design of a synchronous motor control algorithm," *Energies*, vol. 13, no. 16, p. 4057, Aug. 2020.
[35] P. Dini and S. Saponara, "Design of an observer-based architecture and non-linear control algorithm for cogging torque reduction in synchronous motors," *Energies*, vol. 13, no. 8, p. 2077, Apr. 2020.

PIERPAOLO DINI received the M.S. degree in automation engineering and the Ph.D. degree (Hons.) from the University of Pisa. He is a Postdoctoral Researcher with the Department of Information Engineering, University of Pisa. He collaborates in multiple European research projects focused on the development of advanced control and monitoring algorithms for mechatronic systems in industrial applications. His research interests include control systems technology, advanced theory of dynamic systems and control, advanced model-based design paradigms, optimal and nonlinear control, electrical machines and drives, and in-vehicle power electronics.

SERGIO SAPONARA (Senior Member, IEEE) received the master's *(cum laude)* and Ph.D. degrees from the University of Pisa. In 2012, he was a Marie Curie Research Fellow at IMEC. He is a Full Professor of electronics with the University of Pisa. He is an IEEE Distinguished Lecturer and Co-Founder of the Special Interest Group on IoT with the IEEE Circuits and Systems Society (CAS) and the SP Society. He is the Director of I-CAS Laboratory, of Crosslab Industrial IoT, and of the Summer School Enabling Technologies for IoT. He has coauthored more than 300 scientific publications and 18 patents. He is the Leader of many funded projects by EU and by companies like Intel, Magneti Marelli, Ericsson, and PPC. He is an associate editor of several IEEE and Springer Journals.