Delft University of Technology

Delft Center for Systems and Control

Technical report 07-005

Adaptive cruise control for a SMART car: A comparison benchmark for MPC-PWA control methods*

D. Corona and B. De Schutter

If you want to cite this report, please use the following reference instead:
D. Corona and B. De Schutter, "Adaptive cruise control for a SMART car: A comparison benchmark for MPC-PWA control methods," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 365–372, Mar. 2008. doi:10.1109/TCST.2007.908212

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/07_005.html

Adaptive cruise control for a SMART car: A comparison benchmark for MPC-PWA control methods

Daniele Corona and Bart De Schutter

***Abstract*—The design of an adaptive cruise controller for a SMART car, a type of small car, is proposed as a benchmark set-up for several model predictive control methods for nonlinear and piecewise affine systems. Each of these methods has already been applied to specific case studies, different from method to method. This paper has therefore the purpose of implementing and comparing them over a *common benchmark*, allowing to assess the main properties, characteristics and strong/weak points of each method. In the simulations, a realistic model of the SMART, including gear box and engine nonlinearities, is considered. A description of the methods to be compared is presented, and the comparison results are collected in a table. In particular, the trade-offs between complexity and accuracy of the solution, as well as computational aspects are highlighted.**

I. INTRODUCTION

An adaptive cruise controller (ACC) typically aims to increase road safety and passenger comfort. These issues can be modeled by introducing a performance criterion and constraints. This approach is very appealing for several reasons. First, it allows to extend the range of specific design requirements, for instance, fuel consumption and mechanical stress of the vehicle, by simply introducing additional constraints. Secondly, the problem of designing the control law may be naturally cast into a model predictive control (MPC) framework [1], which will result in a constrained minimization problem for which several efficient solvers may be used.

In this paper the design of an ACC for a SMART car is considered as a benchmark problem for existing MPC methods for piecewise affine (PWA) systems. The SMART car is a compact road vehicle produced by the SMART company. In this application the 37 kW gasoline model has been considered. The nonlinear and switching dynamics of the system, as well as the presence of design constraints, make the task of designing an ACC rather challenging, and traditional control techniques may not be suitable. More specifically, the engine torque and the air drag introduce nonlinearities, while the gear box forces the designer to deal with hybrid behavior, which eventually results in PWA models. Part of this paper is hence dedicated to PWA systems, a subclass of *hybrid systems*, i.e., systems exhibiting both continuous (time-driven) and discrete (event-driven) behavior. In particular, a PWA system is composed of a finite set of affine systems and a switching signal that triggers, internally or externally forced, the active mode. PWA models arise, among others, from processes that integrate integer/logical behavior with continuous variables or from quantized inputs [2], or from the linear spline approximation of nonlinearities [3]. The discontinuities, implicitly hidden in their discrete behavior, make the control design a non-trivial task, the complexity of which is additionally increased if constraints are considered. Recently, the control system and computer science communities have been devoted significant efforts to the analysis and control of PWA systems.

Several methods that aim to design the control law for this class were proposed in the literature. Most of them are MPC-based, i.e., the control law that minimizes a finite-horizon performance, is determined based on measurements of the current state of the system and using a model to predict the future behavior, and applied in a receding horizon fashion [4]–[6]. A particular representation of PWA systems that allows to use the MPC scheme is the mixed logical dynamical (MLD) model, for which the control law may be given in implicit [7] or explicit form [8]. Variants that consider robustness [9], [10] or stability properties [11], [12] were also considered. Methods based on the construction of a piecewise Lyapunov function have been developed in [13], [14].

Despite the presence of several methods, an applicative comparison test bed that highlights their main features is, to our best knowledge, missing. The goal of this paper is to propose a benchmark set-up for the MPC on a PWA system, applied to the design of an ACC for a SMART. We implement and compare some of these methods, thus allowing to assess their main properties, characteristics and strong/weak points, for the common ACC case study. In addition, we also include a state of the art version of the ACC used in the automotive industry (based on an adaptive proportional-integral (PI) actuator) in our comparison study.

The paper is organized as follows: we first describe a detailed model of the system, taken from measurements on a real vehicle, and the specific control problem and constraints. Then we provide a short description of eight different control methods, based on PI and MPC in different flavors, namely PWA, nonlinear, on-line and off-line, differing on the level of approximation of the prediction model with respect to the simulation model. The target is to assess and to compare the features of the different control design methods, highlighting the major advantages or disadvantages of the methods. To this

D. Corona and B. De Schutter are with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. B. De Schutter is also with the Marine and Transport Technology department of Delft University of Technology. email: d.corona@tudelft.nl, b@deschutter.info

Research supported by the European NoE HYCON (FP6-IST-511368), the BSIK project TRANSUMO, the Transport Research Center Delft, and STW project 'Model predictive control for hybrid systems' (DMR.5675).

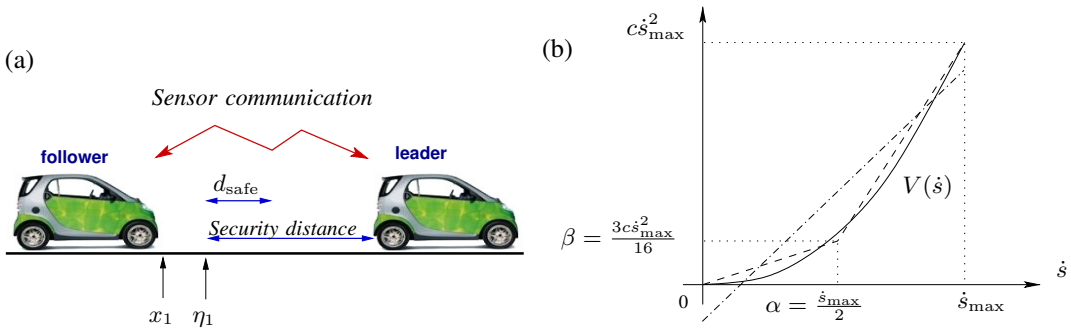

Fig. 1. *(a) ACC set-up and (b) nonlinear friction (solid), PWA approximation (dashed) and affine approximation (dash-dotted).*

purpose we establish a comparison table that highlights key aspects of the control design schemes, the complexity of the mathematical problem, and the quality of the solution.

II. Model and Problem Description

A. Model

The aim of an ACC is to ensure a minimal separation between the vehicles and speed adaptation. In a basic ACC application two cars are driving one after the other (see Figure 1.a). In general platoons of cars can also be considered, see for instance [15], in a multi-agent framework, but here we restrict ourselves to the study of the basic experimental condition of only two vehicles, allowing better insight into the physics of the global system with a reduced number of variables. We assume that the front vehicle communicates its speed and position to the rear vehicle, which has to track them as good as possible. So, for the control design purpose, only the dynamics of the rear vehicle can be considered.

An accurate model of the system considers the air drag proportional to the square of the speed and a constant *road-tire* static friction, proportional to the weight of the vehicle. The dynamics of the rear vehicle are thus described by:

$$m\ddot{s}(t) + (c\dot{s}^2(t) + \mu\, mg)\text{sgn}(\dot{s}(t)) = b(j,\dot{s})u(t) \qquad (1)$$

where $s(t)$ is the position at time t and $b(j,\dot{s})u(t)$ is the traction force, proportional to the normalized throttle/brake position $u(t)$, considered as an input. The mass m of the SMART is equal to 800 kg, the wheel radius R is 0.28 m, the viscous friction coefficient c equals 0.5 kg/m, the Coulomb friction coefficient μ equals 0.01, g is the acceleration due to gravity (9.8 m/s^2), the minimal rotational speed $w_{\min}$ equals 105 rad/s, and the maximal rotational speed $w_{\max}$ is 630 rad/s. The value of the function $\text{sgn}(\dot{s}(t))$ is equal to 1, 0 or -1 when its argument is positive, zero, or negative respectively. The traction force depends on the current gear $j = \{1,\dots,6\}$ and on the ground speed $\dot{s}(t)$. Additionally, we provide the function $b(j,\dot{s})$ in Figure 2, obtained from the transmission ratio of the engine torque curve [16] in the engine rotational velocity range $w \in [w_{\min}, w_{\max}]$: $b(j,\dot{s}) = \frac{T_e(w)p(j)}{R}$, $\dot{s} = \frac{wR}{p(j)}$, where $T_e(w)$ is the engine torque, R is the average radius of the wheels, $p(j)$ represents the gear ratios. Here, we have omitted the dependence on time t of s, w and j. The values of $p(j)$, including also the efficiency of the transmission from engine to wheel, are provided in Table I. Since the maximal engine torque ($T_{e,\max} = 80\,Nm$) may be considered constant [16] in the range $w \in [200, 480]$, we also give the values $b(j)$ in this specific range.

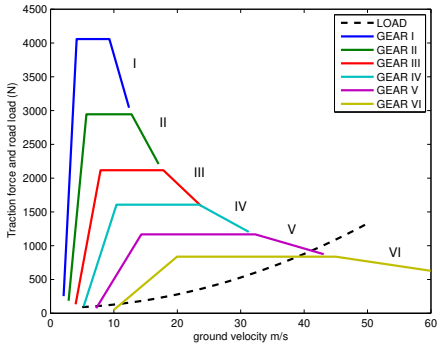

Fig. 2. *Traction force transmitted to the wheel at maximum throttle input for different gears.*

TABLE I
Transmission rates, maximum traction forces, and ground velocity switching conditions in a SMART.

Gear j	Transmission rate $p(j)$	Traction force $b(j)$ (N)	Min. vel. (m/s)	Max. vel. (m/s)
I	14.203	4057	3.94	9.46
II	10.310	2945	5.43	13.04
III	7.407	2116	7.56	18.15
IV	5.625	1607	9.96	23.90
V	4.083	1166	13.70	32.93
VI	2.933	838	19.10	45.84

Braking will be simulated by applying a negative throttle. Due to friction behavior in motion inversion [17], model (1) is valid as long as the ground speed $\dot{s}$ is different from zero. Hence, we impose $\dot{s}$ to be above a nonzero minimum velocity.

A state space representation of system (1) is:

$$\dot{x} = f(x) + B(j,x)u, \qquad (2)$$

with $x \triangleq \begin{bmatrix} s \\ \dot{s} \end{bmatrix}$, $f(x) = \begin{bmatrix} x_2 \\ -\frac{c}{m}x_2^2 - \mu g \end{bmatrix}$, $B(j,x) = \begin{bmatrix} 0 \\ -\frac{b(j,x_2)}{m} \end{bmatrix}$. This model is nonlinear because of the friction and traction forces, and hybrid because of the discrete dependences of b. In the MPC approach we intend to use this model as simulation tool, while using simpler models to make predictions.

TABLE II
Values of the parameters specifying the constraints.

Parameter	Description	Numerical value
$x_{1,\min}$	Min. position	0 m
$x_{1,\max}$	Max. position	3000 m
$x_{2,\min}$	Min. velocity	2.0 m/s
$x_{2,\max}$	Max. velocity	40.0 m/s
d_{safe}	Sec. pos. overshoot	10.0 m
a_{acc}	Comfort acceleration	2.5 m/s^2
a_{dec}	Comfort deceleration	2.0 m/s^2
$u_{\max}$	Max. throttle/brake	1

B. Constraints

Safety, comfort and economy or environmental issues, as well as limitations on the model, result in defining constraints on the behavior of the system. In particular we consider limitations on the state $x = [s, \dot{s}]^{\mathrm{T}}$, i.e., position, velocity, acceleration, and on the control input u. More precisely, we impose that for all $t \geq 0$, we should have $x_{2,\min} \leq x_2(t) \leq x_{2,\max}$, $x_1(t) \leq \eta_1(t) + d_{\text{safe}}$, and $a_{\text{dec}} \leq \ddot{s}(t) \leq a_{\text{acc}}$. These constraints express, respectively, the operational range of the speed, the tracking of the leading vehicle trajectory $\eta = [\eta_1, \eta_2]^{\mathrm{T}}$ within a given tolerance d_{safe} (see Figure 1.a), and bounds on acceleration for comfort or security specifications. We shall consider as well an additional *non-operational* constraint on the position: $x_{1,\min} \leq x_1(t) \leq x_{1,\max}$, which is necessary in the MLD approach of the problem. This constraint is not restrictive, as in an MPC receding horizon approach we can always reset the origin of the position measurements, and let $x_{1,\max}$ be the maximal distance that the vehicle can cover when driving at its maximal speed $x_{2,\max}$ during the entire prediction horizon.

Moreover, we consider limitations on control input: $|u(t)| \leq u_{\max}$, and finally two constraints on the gear shift: $1 \leq j(t) \leq 6$ and $|j(t+dt) - j(t)| \leq 1$, where dt is a finite small time-interval. The last condition forbids jumps of gears with more than one position as these usually provoke non-optimal fuel consumption in up-shifting and mechanical stress in down-shifting. Numerical values are listed in Table II. Although some of these constraints may be violated without causing major damages, i.e., collision or engine breakdown, we decided to consider all of them as *hard*.

Since we are in an MPC framework, we will immediately provide the expression of the constraints in discrete time. Hence, for all k:

$$\begin{gathered} x_{\min} \leq x(k) \leq x_{\max} \\ x_1(k) \leq \eta_1(k) + d_{\text{safe}} \\ a_{\text{dec}} T \leq x_2(k+1) - x_2(k) \leq a_{\text{acc}} T \\ -u_{\max} \leq u(k) \leq u_{\max} \\ 1 \leq j(k) \leq 6 \\ -1 \leq j(k+1) - j(k) \leq 1, \end{gathered} \tag{3}$$

where k is the discrete counter and T is the sampling time.

C. Optimal control problem

The control signal $u(k)$ is designed by solving a constrained finite-horizon optimal control problem, in an MPC receding horizon fashion. In this framework the prediction or acquisition of N_{p} samples ahead of the front vehicle trajectory is used to compute the optimal control law $u(k)$. The MPC approach is largely used to design the control action of constrained systems and in particular PWA systems (see, e.g., [7], [9], [12]). The control action is obtained by solving

$$\begin{aligned} &\min_{\tilde{u}(N_{\text{p}}), \tilde{j}(N_{\text{p}})} J(\theta(k), \tilde{u}(N_{\text{p}}), \tilde{j}(N_{\text{p}})) \triangleq \\ &\sum_{i=1}^{N_{\text{p}}} ||Q_x \varepsilon(k+i)||_1 + ||Q_{\Delta u} \Delta u(k+i-1)||_1 + \\ &\qquad ||Q_{\Delta j} \Delta j(k+i-1)||_1, \end{aligned} \tag{4}$$

subject to the particular prediction model that will be described in the sequel and the constraints derived from physical specifications (see Section II-B). We are interested in minimizing the number of gear switchings Δj, the variation of the control input Δu, and the deviation from a given reference trajectory communicated by the leading vehicle. Here, $\varepsilon(k) \triangleq x(k) - \eta(k)$ is the tracking error, $\tilde{u}(N_{\text{p}}) \triangleq [u(k), \ldots, u(k+N_{\text{p}}-1)]^{\mathrm{T}}$ the sequence of control inputs, $\tilde{j}(N_{\text{p}}) \triangleq [j(k), \ldots, j(k+N_{\text{p}}-1)]^{\mathrm{T}}$ the gear shift sequence, Q_x, $Q_{\Delta u}$ and $Q_{\Delta j}$ are weight matrices of appropriate dimension, $\theta(k)$ is a set of parameters containing the initial conditions and the prediction of the reference trajectory for the next N_{p} sample steps. In this application we have $\theta(k) \triangleq [x(k)^{\mathrm{T}}, u(k-1), j(k-1), \eta(k+1)^{\mathrm{T}}, \ldots, \eta(k+N_{\text{p}})^{\mathrm{T}}]^{\mathrm{T}}$.

Additionally, an appropriately tuned shorter control horizon $N_{\text{c}} < N_{\text{p}}$ may also be considered when we set $\Delta u(k+i) = 0$, $i = N_{\text{c}}, \ldots, N_{\text{p}} - 1$. This has the general advantage of reducing the number of variables and of providing a smoother solution. Nevertheless, here we only consider $N_{\text{p}} = N_{\text{c}}$. The choice of the 1-norm in (4) offers a valid trade-off between the complexity of the optimization problem and the quality of the solution. It allows the use of (mixed-integer) *linear* programming [18]–[20].

We consider a reference trajectory $\eta(k)$ in which the front vehicle is driving at the constant velocity of $15\,m/s$ and its position is obtained by integration of this velocity. This choice permits to study the behavior of the controllers in a smooth driving scenario (extra-urban road with speed limits and a low traffic density) and therefore to compare the features of the different design methods when facing a nominal scenario. More stressful scenarios, i.e., involving complex maneuvers such as abrupt braking or acceleration, may not influence significantly the comparison of the different MPC methods, but they are of major interest for future studies that deal more specifically with the technical design of the controller and especially with the definitions of its safety margins.

In order to solve the problem above, i.e., to design an appropriate control law, we may use a *prediction model* that gives an approximation of the physical system. In an MPC set-up the measured output $x(k)$, possibly affected by disturbances $\Omega(k)$, is plugged into the controller, which also receives the prediction of the reference $\eta(k)$. According to these values, the controller computes the next optimal control input, which is then fed into the real system, or in our case, the full nonlinear simulation model. At the next sampling step, new measurements are obtained and the whole procedure is repeated (i.e., we use a moving or receding horizon approach).

III. Design methods

In the following we propose eight different methods to deal with the nonlinearity raising from the friction force, the engine torque, and the gears. The prediction models and control approaches, extensively described in the sequel, are:

- Nonlinear MPC: NMPC,
- On-line PWA MPC: MLD-on,
- Off-line PWA MPC: MLD-off,
- Gears and linear approximation: GLA,
- Gears and tangent approximation: GTA
- Basic tangent approximation: BTA,
- Basic gain-scheduling approximation: BGS,
- Optimized proportional-integral (PI) controller.

In the first case we consider the exact expression of the friction and implement a nonlinear mixed-integer MPC; in the second case we provide a PWA approximation using least squares splines by the introduction of one breakpoint and then implement a mixed-integer MPC based on the equivalent MLD model. For this particular case an on-line and an off-line solution is calculated. Another possibility is to approximate the friction as $V(\dot{s}) = c\dot{s}^2 + f \simeq c_1\dot{s} + f_1$, ($c_1, f_1$ are chosen using least squares), or to linearize it around the operating point with its tangent. We also take into account methods that are based on very simple prediction models. In these cases we use a linear differential equation where the gear shift action is not considered and the traction force B_j is averaged for every gear and velocity. The nonlinearity due to the air drag is first treated with a tangent around the operating point (in Section III-F), and next gain-scheduled for an off-line method (in Section III-G). The expected advantage over the first five methods is to obtain a rough *good* solution at a very low computational cost, which in many applications may be considered acceptable.

Before proceeding further with the descriptions of the models some additional comments are required for the first five methods regarding the use of gear shift. In all these cases the problem remains, to an extent, hybrid. Moreover, in all methods we approximate the function $b(j, x_2)$ in (2) as follows. We first consider it constant with the velocity and we take, for each gear, its maximum value, as depicted in Figure 2. This yields the six values in Table I, namely, $\{b(1), \dots, b(6)\}$. Then, we define $(\beta_0, \beta_1) \triangleq \arg\min_{\beta_0,\beta_1} \sum_{j=1}^{6} \big(b(j) - (\beta_0 + j\beta_1)\big)^2$.

This allows us to express

$$b(j) \approx b_j \triangleq \beta_0 + j\beta_1 \tag{5}$$

as an affine function of the gear j, with $j \in \{1, \dots, 6\}$. In Figure 3 we depict the approximation of the traction force described above. In order to encode the gear in a binary way, which is necessary to implement an MLD model, at least three binary variables $\delta_i \in \{0, 1\}$, with $i = 1, 2, 3$, are needed. The encoding can be done by setting $j = 1 + \sum_{i=1}^{3} 2^{i-1}\delta_i$, so that to each value of the gear there corresponds *one and only one* logic combination of $\delta_1, \delta_2, \delta_3$, as listed in Table III. Plugging the expression for j into (5) we obtain

$$b_j u = (\beta_0 + \beta_1)u + \beta_1\delta_1 u + 2\beta_1\delta_2 u + 4\beta_1\delta_3 u. \tag{6}$$

TABLE III
Encoding of gear j via three binary variables δ_i.

Gear j	δ_1	δ_2	δ_3
I	0	0	0
II	1	0	0
III	0	1	0
IV	1	1	0
V	0	0	1
VI	1	0	1

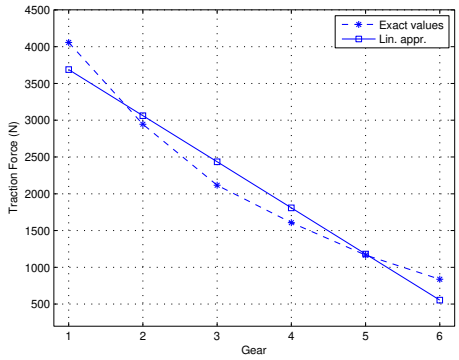

Fig. 3. *Traction force approximation for different gears.*

The gear switching condition is governed by the value of the current velocity. Hence, we have

$$v_L(j) \le \dot{s} \le v_H(j), \tag{7}$$

where j is the current gear position $j \in \{1, \dots, 6\}$ and the values of $v_L(j), v_H(j)$ are given in Table I. Note that the switching condition is not uniquely defined, thus different gears are admitted for a specific value of the speed. The exact modeling of such scenario is possible, but it requires the introduction of several extra binary variables, making the computational aspect of the problem more complex. A simple strategy is to approximate the inequality (7) by

$$v_0 + v_1 j \le \dot{s} \le v_0 + v_1(j+1), \tag{8}$$

which preserves linearity and a one-to-one relation between velocity and current gear. Within this condition the approximation depends only on the choice of the two values v_0 and v_1. The values of v_0 and v_1 are obtained as

$$(v_0, v_1) \triangleq \arg\min_{v_0,v_1} \Big(\gamma_L \sum_{j=1}^{6} \big(v_L(j) - (v_0 + jv_1)\big)^2 + \gamma_H \sum_{j=1}^{6} \big(v_H(j) - (v_0 + (j+1)v_1)\big)^2\Big),$$

subject to $v_0 + v_1 \ge x_{2,\min}$. The choice of the weights γ_L, γ_H was preferred towards the higher velocities ($\gamma_L = 1, \gamma_H = 100$), where the engine works with higher efficiency. We depict this approximation in Figure 4.

A. Method 1: Nonlinear MPC (NMPC)

In this method the prediction model is the discrete-time representation of the simulation model (2). For the integration

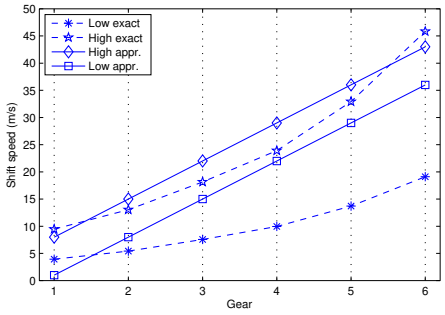

Fig. 4. *Approximation of switching velocities for different gears.*

we use a first-order Euler approximation[1], leading to

$$x(k+1) = x(k) + f(x(k))T + B_j T u(k), \tag{9}$$

where $T = 1\,s$ is the chosen sampling time and $(B_j)_2 \simeq (\beta_0 + \beta_1) + \beta_1\delta_1 + 2\beta_1\delta_2 + 4\beta_1\delta_3$, as in (6).

Using this model, problem (4) is transformed into a *mixed-integer nonlinear* optimization problem (MINLP), of the form

$$J^*(\theta(k)) = \min_{\tilde{y}} f(\tilde{y}) \quad \text{s.t. } g(\tilde{y}) \leq E_\theta \theta(k), \tag{10}$$

where $\tilde{y}$ includes the control variables and some additional dummy variables. The function g and the constant matrix E_θ, represent the feasible area of the optimization problem. In particular they express the constraints on the physical system over the control horizon and on some logic variables appearing in the vector $\tilde{y}$. This problem, to be solved on-line at each step k, can be solved using branch-and-bound algorithms [21], [22]. Note that its complexity is caused by the presence of non-convex constraints and of integer variables.

B. Method 2: Piecewise affine MPC (MLD-on)

A least squares approximation (Figure 1.b) of the nonlinear friction curve $f(x)$ leads to a PWA prediction model:

$$x(k+1) = \begin{cases} A_1 x(k) + F_1 + B_j u(k) & \text{if } x_2(k) < \alpha \\ A_2 x(k) + F_2 + B_j u(k) & \text{if } x_2(k) \geq \alpha, \end{cases} \tag{11}$$

where the matrices A_1, A_2, F_1, F_2 are derived using the data shown in Figure 1.b[2]. To deal with this PWA system we exploit the mixed logical dynamical (MLD) transformation (see [7] and [18, Section 4.3]). This results in the following mixed-integer linear program (MILP):

$$J^*(\theta(k)) = \min_{\tilde{y}} c'\tilde{y} \quad \text{s.t. } E\tilde{y} \leq G + E_\theta \theta(k), \tag{12}$$

where $\tilde{y}$ includes the control variables and some additional dummy variables required to convert the ℓ_1 objective function into a linear one. The linear constraints in (12) include the operational constraints discussed previously, and some additional constraints introduced by the MLD transformation.

[1]In this particular application the error introduced by this approximation versus the exact integration is negligible even for a long simulation time.

[2]For the sake of simplicity we only consider one breakpoint, leading to a PWA composed of two operating modes. A finer approximation is also possible, by setting more than one breakpoint on the nonlinear curve.

C. Method 3: Piecewise affine MPC (MLD-off)

This method is actually a variant of the one described in Section III-B, but it is solved off-line, leading to a *multi-parametric mixed-integer linear program* (mp-MILP). In simple words, problem (12) is solved explicitly in the parameters (there are several algorithms, see for instance [8], [19]). The optimal solution $J^*(\theta)$ and its argument $y^*(\theta)$ are parametrized over θ. Under the conditions given in [8], Theorem 1.16, the functions $J^*(\theta)$ and $y^*(\theta)$ are PWA functions of θ. These coefficients and the corresponding partition of the parameter space can be pre-calculated and stored *off-line*. This strategy avoids solving optimization problems on-line, and the on-line calculations then reduce to the mere search in a look-up table. Although theoretically equivalent to the previous problem, the experiments described in Section IV show that the mp-MILP might introduce numerical difficulties that affect the equivalence of the solution.

D. Method 4: Gears and linear approximation (GLA)

As in the previous section we approximate $f(x)$ with an affine function, leading to the prediction model

$$x(k+1) = A_\ell x(k) + F_\ell + B_j u(k). \tag{13}$$

One possible choice is to obtain matrices A_ℓ, F_ℓ by minimizing the quadratic error between the parabola and the line, as shown in Figure 1.b. The presence of the gear shift keeps this problem mixed-integer, but it differs from the PWA problem because there is one binary variable less. This is quite advantageous if the prediction horizon is short. The transformation into an on-line MILP is obtained by setting $(B_j)_2 u(k) \simeq (\beta_0+\beta_1)u+\beta_1\delta_1 u+2\beta_1\delta_2 u+4\beta_1\delta_3 u$ and considering the additional constraints that convert it into the MLD form. The structure of the MILP is similar to problem (12).

E. Method 5: Gears and tangent approximation (GTA)

Another possible way to linearize the friction nonlinearity is to use as a prediction model the affine system tangent to the current operating point [23]. This idea is actually very efficient for smooth nonlinear systems with a relatively small sampling time. As in the previous section we approximate $f(x)$ with an affine function, with a slope equal to the derivative of the friction curve around the current velocity. This gives

$$x(k+1) = A_\tau(x(k))x(k) + F_\tau(x(k)) + B_j u(k). \tag{14}$$

The transformation into an on-line MILP is obtained by setting $(B_j)_2 u(k) \simeq (\beta_0 + \beta_1)u + \beta_1\delta_1 u + 2\beta_1\delta_2 u + 4\beta_1\delta_3 u$ and considering the additional constraints that convert it into the MLD form. The structure of the MILP is similar to problem (12).

F. Method 6: Basic tangent approximation (BTA)

This prediction model neglects the presence of the gear shift. In other words we do not assume the traction force, expressed by the coefficient B_j as dependent from the current gear or the current velocity. Hence, the prediction model is

$$x(k+1) = A_\tau(x(k))x(k) + F_\tau(x(k)) + Bu(k), \tag{15}$$

where the coefficient B is obtained as an average of the coefficients listed in Table I. The rough approximation has the clear advantage of leading to an on-line *linear* optimization problem of the form

$$J^*(\theta(k)) = \min_{\tilde{y}} c'\tilde{y} \quad \text{s.t. } E\tilde{y} \leq G + E_\theta\theta(k), \tag{16}$$

the complexity of which is polynomial (fast), unlike previous problems, which are typically NP-hard. The value of the gear shift in this case is chosen according to the value of the current velocity and (8).

G. Method 7: Basic gain-scheduling approximation (BGS)

The previous method also suggests an off-line version, in a gain scheduling fashion. The nonlinear curve depicted in Figure 1.b, is approximated into, say, $M = 6$ linear models m_1, m_2, m_3, m_4, m_5, m_6 in point to point secant approximation. For each affine model m_i we solve an off-line mp-LP [24], [25] problem of the form (16). More precisely we construct $M = 6$ look-up tables, each valid for a given range of velocity. In the simulation the controller selects the table according to the current value of the speed. As in the previous method the gear is chosen based on the velocity range.

H. Method 8: Proportional-integral action (PI)

As additional method we implement a proportional-integral (PI) controller. This is the technique mostly used in practice [26]. The controller first computes a desired acceleration

$$a_{\rm d}(k) = k_{\rm I}\varepsilon_1(k) + k_{\rm P}\varepsilon_2(k), \tag{17}$$

where $k_{\rm P}$ and $k_{\rm I}$ are the proportional and integral coefficients and $\varepsilon(k) = x(k) - \eta(k)$ is the tracking error at step k. Then the actuators regulate the throttle, the gear and the braking action in order to better achieve the desired value of the acceleration.

In industrial versions of the device as used for ACC the coefficients $k_{\rm P}$, $k_{\rm I}$ depend on the current value of the state $x(k)$ (position and velocity) and of the tracking error signal $\varepsilon(k)$, according to a specifically designed bell-shaped curves [26]. The parameters of these curves (offset and peak values and standard deviation) are tuned empirically to obtain high comfort in acceleration and high security in braking for a variety of scenarios. In this study we have tuned the mentioned parameters so that the controller minimizes the performance index described in Section II-C for the given tracking scenario.

IV. Numerical results

All methods were implemented in `Matlab 7` on an `INTEL Pentium 4, 3 GHz` processor. All optimizations, LP and MILP, were performed with `Cplex` under `TOMLAB v5.1`; the *multi-parametric* problems (methods MLD-off and BGS) are solved with the `multi-parametric toolbox MPT v2.6` [20]. The MINLP (*mixed-integer nonlinear program*) of method NMPC (Section III-A) is solved with the *Branch-and-Bound* algorithm of `TOMLAB v5.1`, toolbox `MINLP v1.5`, and the optimal coefficients for method PI are obtained via the nonlinear programming function `fmincon` of the Matlab optimization toolbox.

A. General experimental set-up

The experiments, carried out in computer simulation, allowed us to establish the comparison issues among the different methods described previously. Additionally, they exhibit a positive and encouraging motivation to perform a real-life emulation. It should however be remarked that, for a possible embedded solution in a real SMART, several technical issues should be regarded, like the sensor system, the resources of the on-board electronics, the real-life disturbances and the actuators delays. The cost of the device is also a relevant discrimination parameter. Note that modern technology (differential GPS, laser sensors and extended Kalman filters [27]) provides fast and highly accurate measurements, with a maximal error of 1 m in positioning and 0.1 m/s in velocity.

The general data common to all experiments are as follows: we have taken $Q_x = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}$, $Q_{\Delta u} = 0.1$, $Q_{\Delta j} = 0.01$, $N_{\rm p} = N_{\rm c} = 2$, $T = 1\,s$, a simulation time of $75\,s$, throttle initial position equal to 0, initial gear I, and initial state $[0, 5]^{\rm T}$. The choice of the weight matrices strongly penalizes the gap between reference and vehicle position compared to the other variables. In these experiments the reference (the leading vehicle) is moving with a constant speed of 15 m/s, (54 km/h). The controller measures its current state, receives the reference state, and *predicts*[3] the reference in the subsequent $N_{\rm p} - 1$ future samples. On the basis of previous gear and control input information it evaluates the optimal decision strategy. In the on-line methods this is done by solving an optimal control problem, in the off-line methods by consulting a pre-scheduled table.

The integration of (1) is done after the optimization, using the `Matlab ode45` subroutine and assuming the input u, j constant.

B. Points of comparison and results

The comparison topics are listed in Table IV, and for each line of the table the worst entry is indicated in **bold** and the best in *italics*. The comparison is divided into four groups.

The first one (*computational features*) refers to strictly computational highlights of the problem, and should orientate the reader with time and memory demands and complexity of the method. We use the acronyms NP-H and P to indicate NP-hard and Polynomial complexity. For what concerns the on-line computational time the *maximum* and *average* values along the whole simulation time are collected. Linear and off-line methods (BTA, BGS, MLD-off and PI) are really competitive compared to the others, especially with the method NMPC. As a drawback the off-line methods require a longer off-line pre-computation. We remark here that the sampling time $T = 1\,s$ is longer than in common ACC devices, where measurements are taken at the frequency of 5 to 10 Hz [1], [29] (that is $T = 0.1 - 0.2\,s$). Nevertheless, this is not restrictive; in fact all methods (except for NMPC) require an on-line computation time shorter than 0.1 s.

[3]If the leading vehicle is human driven, it is not useful to predict the reference over a long future period. Hence, we have limited the prediction period to $N_{\rm p} = 2$. If automatically driven vehicles [28] are used, then higher values may be selected for $N_{\rm p}$.

The major advantage of the off-line methods is that they *do not* require the optimizer on-board, but merely an efficient data-base browser. In a real-life application this is highly preferable, since the performance of an on-board platform is unquestionably poorer than that of a desktop computer. Moreover, the optimizers require extra on-board memory (indicated in Table IV with '+opt.') and may have a cost impact due to software licenses. On the other hand, off-line MPC methods require a bigger on-line memory. Under these considerations the method 8 (PI) is highly competitive, as it does not require a significant amount of on-line memory.

The *Max tractable* N_{p}, only applicable for MPC methods, is the biggest N_{p} such that the on-line computational time is smaller than the sampling time $T = 1\,s$. For the MPC off-line methods, this value is the biggest N_{p} such that the required on-line memory is smaller than $128\ Mb$, the memory capacity of an on-board chip.

Finally, the item *Number (#) of regions* (for off-line MPC methods) is an indicator of the granularity of the solution: when integers are involved the look-up table is more complex.

The second group of comparison points refers to the *programming features*, such as basic data of the corresponding optimization problem, and in particular the size of the problem. The *number of variables* (real and integer), the *number of constraints* (linear and nonlinear), and the number of parameters (i.e., the dimension of $\theta(k)$) which affects the complexity for the off-line methods, are computed. Methods 1 to 5, which make use of the more complex gear shift prediction model, have a very high number of variables. This is due to the transformation of the problem into an equivalent one, as it happens in particular for the MLD-on method which requires the introduction of several auxiliary variables and constraints. This results in higher computation time and memory requirements. In this section of the table we also recall whether the method is on-line (*Y = 'Yes'*).

The third group of the table lists some important features of the quality of the solution, providing a better insight into the physical/mechanical aspects of the problem. The first indicator is the total cost of the evolution in closed loop. A higher value of the cost means, broadly speaking, a worse tracking of the position. For this item the most approximate methods behave better. On the counterpart, it can be seen in the following line, they violate the constraint on the acceleration, due to a very aggressive initial action. The PI controller, which does not allow to include constraints, performs the poorest. Other aspects are also listed, in particular the maximum and minimum Δu, namely the variation of the throttle or brake position. All methods behave quite similarly for this item, due to the fact that they all exhibit an initial effort to reach the target: in this case a longer horizon would produce some differences. Next we consider transient features: position and velocity overshoot, the duration of the transient on the velocity tracking[4] and the number of gear switchings made to reach the steady state of the velocity. In particular with *position overshoot* we indicate with how many meters the vehicle overtakes the reference[5]. In all cases the linear methods are really competitive.

The same conclusion cannot be drawn for the number of constraint violations, in the fourth group of the table: in this case the bigger model mismatch of the linear methods compared to the MLD or NMPC methods is the source of numerous constraint violations. This shows once more the importance of the trade-off in the MPC framework between the accuracy of the prediction model and the quality of the solution. To better highlight this aspect, the same computations were performed in the presence of disturbances. In particular two cases are reported: measurement errors (abbreviated *dist.*) on position and velocity (uniformly random distributed error of $1\,m$ for the position and $0.1\,m/s$ for the speed) and model variation (abbreviated *mdl. var.*). For the former case the number of gear switchings is *unstable* for method GLA (28 switchings) and PI (38 switchings), while the other methods are not affected. In the latter case a particular scenario with wet asphalt (smaller friction coefficient $\mu = 0.005$), loaded vehicle (higher mass $m = 900\,kg$), long driving (higher tire pressure and bigger wheel radius $R = 0.30\,m$) is depicted. As expected in this case the on-line methods are not affected (they recompute on-line the optimization) but the look-up table or pre-computed coefficients for the PI, generated with nominal parameters, will only suggest sub-optimal solutions and possibly more constraint violations.

V. Conclusions

We have presented a benchmark that serves as test bed to compare MPC-based control methods developed for PWA systems. More specifically, we have considered the design an adaptive cruise controller for a SMART, and we have considered seven different variants (on-line and off-line), with different degrees of approximation of the friction and of the prediction model. In addition, we have considered a version of an ACC controller as it is used in industry (based on an adaptive PI method). We have compared and assessed the different methods including the trade-offs between performance and computational aspects. The results are collected in a table from which it is possible to recognize the expected behavior of the different methods and which allows to compare the strong and weak points of each of the methods.

Topics for future research include: considering more complex scenarios, performing the comparison on real vehicles, including additional controllers in the comparison, and investigating whether the obtained results also apply to other applications.

References

[1] V. Bageshwar, W. Garrard, and R. Rajamani, "Model predictive control of transitional maneuvers for adaptive cruise controller," *IEEE Trans. Vehicular Technology*, vol. 53, no. 5, pp. 1573–1585, Sep. 2004.

[2] N. Elia and S. Mitter, "Quantization of linear systems," in *Proc. 38th IEEE Conf. on Dec. and Contr.*, Phoenix, Arizona USA, Dec. 1999, pp. 3428–3435.

[3] E. Sontag, "Nonlinear regulation: the piecewise affine approach," *IEEE Trans. Automatic Contr.*, vol. 26, no. 2, pp. 346–357, Apr. 1981.

[4] The time required by the controller to keep the velocity within a 5 % band around the reference.

[5] Note that the hard constraint $x_1(k) \leq \eta_1(k) + d_{\mathrm{safe}}$ is still satisfied.

Method	NMPC	MLD-on	MLD-off	GLA	GTA	BTA	BGS	PI
Computational features								
Complexity	NP-H	NP-H	NP-H	NP-H	NP-H	P	NP-H	NP
Max on-line time (s)	**0.52**	0.0521	0.0074	0.051	0.052	0.036	0.00048	*0.00019*
Avg. on-line time (s)	**0.33**	0.0478	0.0052	0.042	0.042	0.035	*0.00007*	0.00013
Off-line time (s)	*0.035*	0.057	3480	0.053	0.042	0.051	630.52	$\mathbf{2.11 \times 10^4}$
On-line mem. (Mb)	0.33+opt.	0.46+opt.	16.6	0.36+opt.	0.36+opt.	0.22+opt.	4.09	~ 0
Off-line mem. (Mb)	0.05	0.09	0.46+opt.	0.06	0.05	0.004	0.08+opt.	~ 0
Max tractable $N_{\rm p}$	**2**	10	3	29	31	*>70*	5	–
# regions	–	–	**2704**	–	–	–	*630*	–
Programming features								
Program class	NMPC	MILP	mpMILP	MILP	LP	LP	mpLP	NLP
On-line method	Y	Y	N	Y	Y	Y	N	N
# variables	26	**32**	**32**	26	26	*8*	*8*	9
# constraints	82	**102**	**102**	86	86	*44*	*44*	–
# nonl. constr.	**2**	–	–	–	–	–	–	–
# binary vars.	6	**8**	**8**	6	6	–	–	–
# parameters	–	–	**10**	–	–	–	*7*	–
Solution features								
Cost of evolution	132.65	120.97	116.56	138.35	**142.65**	*60.68*	73.62	104.91
Max acc. (m/s^2)	*2.34*	2.46	2.46	2.42	2.39	3.5	3.5	**4.93**
Max dec. (m/s^2)	1.86	1.79	1.79	1.83	1.86	1.38	*1.35*	**2.89**
Max $\Delta u(k)$	1.16	*0.73*	1.18	0.78	1.16	1.0	0.99	**1.5**
Min $\Delta u(k)$	-1.53	-1.06	-1.12	-1.24	-1.53	*-1.01*	-1.07	**-2**
Pos. overshoot (m)	6.58	5.08	4.18	**7.68**	7.56	5.84	5.89	*0.5*
Vel. overshoot (m/s)	6.4	5.98	5.8	6.49	**6.7**	*3.22*	3.3	6.24
Transient at 5% (s)	16	15	15	**17**	**17**	*6*	*6*	10
# gear-switches	**6**	**6**	4	**6**	**6**	*2*	*2*	4
Effect of disturbances								
# violations	*0*	*0*	*0*	*0*	*0*	3	3	**6**
# violations (dist.)	*0*	2	2	1	*0*	3	3	**27**
# violations (mdl. var.)	*0*	*0*	2	*0*	*0*	3	4	**6**
# gear-switches (dist.)	6	6	4	28	6	*2*	*2*	**38**

TABLE IV

Benchmark problem: points of comparison for the 8 methods described in Section III with $N_{\rm p} = 2$ for MPC.

[4] E. Camacho and C. Bordons, *Model Predictive Control*. London: Springer-Verlag, 1998.

[5] J. Maciejowski, *Predictive Control with Constraints*. Harlow, England: Prentice-Hall, 2002.

[6] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.

[7] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, Mar. 1999.

[8] F. Borrelli, *Constrained Optimal Control of Linear and Hybrid Systems*, ser. LNCIS 290. Berlin: Springer–Verlag, 2003.

[9] E. Kerrigan and D. Mayne, "Optimal control of constrained, piecewise affine systems with bounded disturbances," in *Proc. 41th IEEE Conf. on Dec. and Contr.*, Las Vegas, USA, Dec. 2002, pp. 1552–1557.

[10] S. Raković, E. Kerrigan, and D. Mayne, "Optimal control of constrained piecewise affine systems with state and input–dependent disturbances," in *Proc. Math. Theory of Networks and Sys.*, Leuven, Belgium, Jul. 2004.

[11] G. Ferrari-Trecate, F. Cuzzola, D. Mignone, and M. Morari, "Analysis of discrete-time piecwise affine and hybrid systems," *Automatica*, vol. 38, no. 12, pp. 2139–2146, Dec. 2002.

[12] M. Lazar, W. Heemels, S. Weiland, A. Bemporad, and O. Pastravanu, "Infinity norms as Lyapunov functions for model predictive control of constrained PWA systems," in *Hybrid Systems: Computation and Control*, ser. LNCS, no. 3414. Zürich, Switzerland: Springer Verlag, 2005, pp. 417–432.

[13] S. Hedlund and A. Rantzer, "Optimal control of hybrid systems," in *Proc. 38th IEEE Conf. on Dec. and Contr.*, Phoenix, USA, Dec. 1999, pp. 3972–3976.

[14] M. Johansson, *Piecewise Linear Control Systems*, ser. LNCIS 284. Berlin: Springer–Verlag, 2003.

[15] D. Godbole and J. Lygeros, "Longitudinal control of the lead car of a platoon," *IEEE Trans. Vehicular Technology*, vol. 43, no. 4, pp. 1125–1135, Nov. 1994.

[16] Smart Website, "`http://www.smart-training-online.com/`."

[17] F. Gustafsson, "Slip–based tire–road friction estimation," *Automatica*, vol. 33, no. 6, pp. 1087–1099, Jun. 1997.

[18] A. Bemporad, *Hybrid Toolbox–User's Guide*, Siena, 2003, available from `http://www.dii.unisi.it/hybrid/toolbox`.

[19] V. Dua and E. Pistikopoulos, "An algorithm for the solution of multi-parametric mixed integer linear programming problems," *Annals of Op. Research*, vol. 99, no. 1–4, pp. 123–139, Dec. 2000.

[20] M. Kvasnica, P. Grieder, M. Baotić, and F. Christophersen, *Multi-Parametric Toolbox MPT: User's Manual*, (ETH) Zurich, Jun. 2004, Available from `http://control.ee.ethz.ch/~mpt`.

[21] R. Fletcher and S. Leyffer, "Solving mixed integer nonlinear programs by outer approximation," *Mathematical Programming*, vol. 66, no. 1-3, pp. 327–349, Aug. 1994.

[22] C. Floudas, *Nonlinear and Mixed-Integer Optimization*, ser. Topics in Chemical Engineering. New York: Oxford University press, 1995.

[23] A. Beccuti, T. Geyer, and M. Morari, "A hybrid system approach to power systems voltage control," in *Proc. 44th IEEE Conf. on Dec. and Contr.*, Seville, Spain, Dec. 2005, pp. 6774–6779.

[24] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, Jan. 2002.

[25] A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming–the explicit solution," *IEEE Trans. Automatic Contr.*, vol. 47, no. 12, pp. 1974–1985, Dec. 2002.

[26] D. Yanakiev and I. Kanellakopoulos, "Nonlinear spacing policies for automated heavy-duty vehicles," *IEEE Trans. Vehicular Technology*, vol. 47, no. 4, pp. 1365–1377, Nov. 1998.

[27] R. Hallouzi, V. Verdult, H. Hellendoorn, and J. Ploeg, "Experimental evaluation of a co-operative driving set-up based on inter-vehicle communication," in *Proc. IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, Jul. 2004.

[28] P. Ioannou and C. Chien, "Autonomous intelligent cruise control," *IEEE Trans. Vehicular Technology*, vol. 42, no. 4, pp. 657–672, Nov. 1993.

[29] G. Marsden, M. McDonald, and M. Brackstone, "Towards an understanding of adaptive cruise control," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 1, pp. 33–51, Feb. 2001.