Technical report 06-008

Adaptive cruise controller design: A comparative assessment for PWA systems

D. Corona and B. De Schutter

If you want to cite this report, please use the following reference instead:
ADAPTIVE CRUISE CONTROLLER DESIGN: A COMPARATIVE ASSESSMENT FOR PWA SYSTEMS

Daniele Corona∗ Bart De Schutter∗

∗ Delft Center for System and Control, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands

Abstract: We propose the design of an adaptive cruise controller (ACC) of a Smart vehicle as a benchmark set up for methods developed for piecewise affine (PWA) systems based on model predictive control (MPC) arguments. The control law of the system aims at achieving a trade-off between tracking and fuel consumption while guaranteeing specific constraints, related to physical limitations, safety/comfort issues, environment protection and energy saving. In this paper we consider some PWA MPC control design methods, an on-line, an off-line and a robust on-line and compare them with an on-line linear approximation and an off-line gain scheduling approach. The results will be briefly described and the algorithms will be tested in terms of key issues for real implementation and accuracy of the solution.

Keywords: piecewise affine systems, model predictive control, engine control, multi-parametric mixed integer linear programming.

1. INTRODUCTION

Recently, significant efforts of the control system and computer science communities have been devoted to the study and design of hybrid systems. Within this class particular attention was paid to piecewise affine (PWA) systems, namely, a finite set of affine system and a switching signal that switches, internally or externally forced, from one affine mode to another.

PWA systems arise from modeling processes that integrate integer/logical behavior with continuous variables or from quantized inputs (Elia and Mitter, 1999), or from the linear spline approximation of nonlinearities (Sontag, 1981). The discontinuities implicitly hidden in the discrete behavior of these systems make the control design a nontrivial task, the complexity of which is additionally increased if constraints are considered.

Nevertheless several methods that aim to design the control law for this class were proposed in the literature and some of them are MPC based. An equivalent model, the mixed logical dynamical (MLD) model (Bemporad and Morari, 1999), is used to compute the control law implicitly or
explicitly (Borrelli, 2003). Several variants that consider robustness (Kerrigan and Mayne, 2002) or stability properties (Lazar et al., 2005) were also considered. Methods based on the construction of a piecewise Lyapunov function have been developed in (Hedlund and Rantzer, 1999).

Despite the presence of several methods, an applicative comparison test bed that highlights their main features is, to our best knowledge, missing. Therefore in this paper a benchmark set up for the design of MPC for a PWA system is proposed, arising from an application that is focused on the environmental issues, as well as limitations on the state, the tracking of the leading vehicle trajectory and jerk. We have, \( \forall k \), for the control input:

\[
|u(k)| \leq u_{\text{max}},
\]

Numerical values are listed in Table 2. Note that 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.

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 breakpoints on the nonlinear curve.

### Table 1. Entries of equation (1).

| \( m \): Mass of vehicle | 800 kg |
| \( c \): Viscous coefficient | 0.5 kg/m |
| \( \mu \): Friction coefficient (dry asphalt) | 0.01 |
| \( b \): Traction force | 3700 N |
| \( g \): Gravity acceleration | 9.8 m/s^2 |
| \( \alpha \): Switching velocity | 18.75 m/s |

### Table 2. Values of the constraints.

| \( x_{2,\text{min}} \): Min. velocity | 5.0 m/s |
| \( x_{2,\text{max}} \): Max. velocity | 37.5 m/s |
| \( d_{\text{safe}} \): Max. position overshoot | 5.0 m |
| \( a_{\text{acc}} \): Max. acceleration | 2.5 m/s^2 |
| \( a_{\text{dec}} \): Min. acceleration | -1 m/s^2 |
| \( \xi \): Comfort jerk | 2.0 m/s^3 |
| \( u_{\text{max}} \): Max. throttle/brake | 1 |
| \( \Delta u \): Max. throttle/brake variation | 0.2 |

Mode \( i = 1(2) \) is active when \( \dot{s}(t) < (\geq) \alpha \). The coefficients \( c_i, f_i \) are derived using the data shown in Figure 1.b.

The discrete-time state-space representation (sampling time \( T = 1s \), zero order hold) is

\[
x(k+1) = A_i x(k) + B_i u(k) + F_i, \quad x_2(k) < \alpha \quad (2)
\]

with

\[
A_1 = \begin{bmatrix} 1 & 0.97 \\ 0 & 0.99 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 2.31 \\ 4.61 \end{bmatrix}, \quad F_1 = \begin{bmatrix} 0.05 \\ 0.10 \end{bmatrix}, \\
A_2 = \begin{bmatrix} 1 & 0.98 \\ 0 & 0.96 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 2.28 \\ 4.54 \end{bmatrix}, \quad F_2 = \begin{bmatrix} 0.22 \\ 0.44 \end{bmatrix},
\]

where \( x_1(k) \) is the position and \( x_2(k) \) is the velocity of the rear vehicle.

**Constraints.** Safety, comfort and economy or environmental issues, as well as limitations on the model, constrain the performance of the system. Since some methods require bounded variables we assume a minimum \( (x_{1,\text{min}} = 0 m) \) and a maximum \( (x_{1,\text{max}} = 2000 m) \) position. In particular we consider limitations on the state \( x(k) \) and on the input \( u(k) \). More precisely, \( \forall k \),

\[
x_{\text{min}} \leq x(k) \leq x_{\text{max}} \\
x_1(k) \leq \eta_1(k) + d_{\text{safe}} \\
a_{\text{dec}} T \leq x_2(k+1) - x_2(k) \leq a_{\text{acc}} T \\
| x_2(k+1) - 2x_2(k) + x_2(k-1) | \leq \xi T^2.
\]

The above equations express, respectively, the operative range of the state, the tracking of the leading vehicle trajectory \( \eta(k) = [\eta_1(k), \eta_2(k)]^T \) within a given overshoot \( d_{\text{safe}} \), bounds on acceleration and jerk. We have, \( \forall k \), for the control input:

\[
|u(k)| \leq u_{\text{max}}, \quad |u(k+1) - u(k)| \leq \Delta u.
\]
In this section we describe the design methods (Borrelli, 2003).

\[
\min J(\theta(k), \tilde{u}(N_p)) \triangleq \sum_{j=1}^{N_p} ||Q \tilde{e}(k+j)||_1 + ||Ru(k+j-1)||_1,
\]
s.t., (2), (3) and (4). Here \(\tilde{e}(k+j) = x(k+j)-\eta(k+j)\) is the tracking error, \(\tilde{u}(N_p) = [u(k), \ldots, u(k+N_p-1)]\) the sequence of control inputs, \(Q, R\) are weight matrices, \(\theta(k) \triangleq [u(k-1), z(k-1)^T, x(k)^T, \eta(k+1)^T, \ldots, \eta(k+N_p)^T]^T\) is a set of parameters. In this framework the front vehicle communicates the prediction of \(N_p\) samples ahead of its trajectory.

Note that an appropriately tuned shorter control horizon \(N_c < N_p\) may also be studied, i.e., \(u(k+j) = u(k+N_c-1), j = N_c, \ldots, N_p - 1\). This choice has the general advantage of reducing the number of variables and of providing a smoother solution. Nevertheless here we only consider \(N_p = N_c\) because a tuning issue was beyond the scope of this paper. The \(\ell_1\)-norm allows the use of (mixed integer) linear programming and usually results in better performance than the \(\ell_{\infty}\)-norm (Borrelli, 2003).

3. DESIGN METHODS

In this section we describe the design methods that we have used on the problem above. For all methods we employed the weight matrices \(Q = \text{diag}(0.8, 0.1)\) and \(R = 0.01\).

Since some of these methods require the transformation of the PWA system into a mixed logical dynamical (MLD) form, we show for completeness how this was performed in our setup.

**PWA to MLD transformation.** The PWA system (2) is transformed into an MLD system by the introduction of a binary variable \(\delta(k)\) (Bemporad and Morari, 1999). The value of \(\delta(k)\) equals 0(1) when the active mode is system 1(2). Hence the new model of the system is:

\[
x(k+1) = A_1x(k) + Le(k) + F_1, \quad (6)
\]

where \(L = [A_2 - A_1 | B_2 - B_1 | F_2 - F_1 | B_1]\) and \(v(k) = [z(k)^T, y(k), \delta(k), u(k)^T]\) (with \(z(k) = x(k)\delta(k), y(k) = u(k)\delta(k)\), \(\delta(k) \in [0, 1]\)) is the auxiliary mixed logical control input. The variables \(z(k), y(k)\) are nonlinear, but they can be converted into equivalent mixed-integer linear inequalities (Bemporad and Morari, 1999):

\[
-x_{\min}(1 - \delta(k)) \leq z(k) - x_{\max}(1 - \delta(k)) \leq -x_{\min}(1 - \delta(k))
\]

\[
-y_{\max}(1 - \delta(k)) \leq y(k) - u_{\max}(1 - \delta(k)) \leq y_{\max}(1 - \delta(k)).
\]

The switching condition leads to

\[
-\delta(k)(v_{\min} - \alpha) \leq x_2(k) - v_{\min}
\]

\[
\delta(k)(\alpha - v_{\max}) \leq -x_2(k) + \alpha.
\]

**Method 1: on-line direct approach.** This method (Bemporad and Morari, 1999) solves problem (5) s.t. the MLD model (6), the additional constraints (7) and (8). This is appropriately converted into a mixed integer linear program (MILP) as it follows:

\[
J(\theta(k))^* = \min_{\tilde{y}} c^T \tilde{y}
\]

s.t. \(E \tilde{y} \leq G + E_0 \theta(k)\),

where the variable \(\tilde{y}\) includes the prediction of the control variable \(\tilde{v}\) and the dummy variables introduced to convert the \(\ell_1\) problem into a linear problem. In a receding horizon fashion, the problem is solved on-line and the best control action is computed at step \(k\). At the next step \(k+1\) the set of parameters is updated and a new optimization problem is formulated.

**Method 2: off-line explicit approach.** Problem (9) can be solved parametrically, as proposed in (Bemporad et al., 2000), based on the theoretical results described in (Dua and Pistikopoulos, 2000). This leads to an mp-MILP (multi parametric mixed integer linear programming) as described in (Borrelli, 2003), Chapter 8.
The mp-MILP solver provides a partition of convex polyhedra in the parameter space \( \theta \), and a PWA control law of the form \( \tilde{y}(\theta) = F_i \theta + G_i \), that serves as a look-up table during the real time evolution. The software we used for this case is part of the Multi-Parametric-Toolbox release 2.5 (Kvasnica et al., 2004).

**Method 3: on-line linear approximation.** In alternative to the hybrid optimization we also considered a locally linear approximation approach. The method consists in using a locally linearized prediction model. As detailed in (Becchi et al., 2005), this technique is appealing for relatively slow processes. The manipulated variables are computed on the base of the linear model, that approximates the nonlinear friction force \( V = cv^2 \) with its tangent line, i.e., \( V = cv^2 \approx 2cv_0v - cv_0^2 \).

**Method 4: off-line linear approximation.**

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 \( M = 4 \) affine models \( m_1, m_2, m_3, m_4 \) in point to point secant approximation, as illustrated in Figure 2.

For each affine model \( m_i \) we solve an off-line mp-LP problem of the form (9). More precisely we construct \( M = 4 \) 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.

**Method 5: Positively invariant set.** This on-line approach (Lazar et al., 2005) is aimed to determine a positively invariant set \( X_{N_p} \) for the PWA system, within which a strictly linear control action may be used. The cost function of the form

\[
J(\theta(k), \bar{u}(N_p)) \triangleq \sum_{j=1}^{N_p} ||Q_j z(k+j)||_1 + ||Ru(k+j-1)||_1,
\]

where \( Q_j = Q \quad \forall j = 1, \ldots, N_p - 1 \), is employed as a candidate Lyapunov function to construct (a) an additional end point constraint and (b) a terminal cost matrix \( Q_{N_p} \). The method (Lazar et al., 2005) is conceived for the \( \ell_\infty \)-norm, but it can be easily adapted to the \( \ell_1 \)-norm.

The reformulation of problem (5) with the additional conditions (a) and (b) has the important advantage of providing a stabilizing control law for the hybrid system. Hence we compute an on-line MPC controller over the MLD model (6), enriched with an end point constraint.

In this specific tracking problem, the end point constraint (Figure 3) computed off-line, is centered in the \( N_p \)-th point of the reference trajectory. The terminal cost matrix is \( Q_{N_p} = \begin{bmatrix} 4.58 & 0.45 \\ 5.14 & 4.15 \end{bmatrix} \).

### 4. COMPARISON ISSUES AND RESULTS

The five methods are implemented in Matlab 7, Linux 2.4.22 OS on an INTEL Pentium 4, 3GHz. All optimizations, LP and MILP, are performed via the solvers embedded in TOMLAB v5.1, except for method 2 (CDD Criss Cross solver).

The prediction model for methods 3 and 4 is a linear approximation and an MLD model for methods 1, 2 and 5. Simulation is carried out over the nonlinear continuous time system (1). Methods 3 and 4 are really competitive as long as the system and prediction model are close. On the other hand, methods 1, 2 and 5, based on PWA approximation of the nonlinearity, are not significantly affected by the model mismatch. The integration of equation (1) is done with Matlab ode45 with relative and absolute tolerances \( 1.0 \times 10^{-8} \), \( \text{max step size} \ 1.0 \times 10^{-3} \).

We have considered two references: (a) a smooth trajectory, with constant velocity and (b) irregular trajectory, with variable velocity, depicted in Figure 4. Some other features of the common setup

---

Footnote: 3 The sampling time plays a fundamental role in this specific application.
that off-line methods do not require the optimizer on board, which can be relevant in an integrated design that also takes into account budget issues.

We also provide the number of times when infeasibility occurred during the simulation period. This index is related to the length of the prediction horizon, and it should decrease as the value of $N_p$ increases. The next row of the tables report the cost of the evolution of the obtained control law.

For the on-line methods the item Maximum tractable $N_p$ is obtained by choosing the smallest $N_p$ for which the computational time required by the optimizer is lower than the sampling time and the solution is 100% feasible for the whole simulation period. We observe that this is possible for PWA approximations, but not for methods 3 and 4, for which increasing the prediction horizon does not provide significant improvement on the feasibility of the solution.

Method 5 contains an additional terminal constraint, and it is completely infeasible for low values of $N_p$. The minimum $N_p$ that guarantees 100% feasibility is $N_p = 17(19)$ (reference b), but unfortunately for this size the computation time becomes longer than the sample time ($T = 1$ s).

For the off-line methods, namely 2 and 4, the value of Maximum tractable $N_p$ is based on reasonable on-line computational time and data storage requirements. As it is shown in Tables 4 and 5 the look-up process to the partition table requires more time than the sampling time $T$. Nevertheless, the look-up process may be improved by use of appropriate binary search algorithms (Tøndel et al., 2003), the complexity of which is logarithmic.

One major limitation of off-line methods is that the look-up table is only valid for a specific configuration of the system. Thus these methods are not able to handle possible variation of the system parameters values, and a recomputation of the table may be required.

5. CONCLUSIONS

Some methods conceived for computing control laws of PWA systems were used to design an ACC for a Smart vehicle. We have made a comparison among several methods. In particular, methods based on MLD transformation (1 and 5) appear to be very reliable and accurate although they require to perform an on-line optimization. On the other side, off-line methods seem to be more efficient in terms of feasibility, but they suffer of high complexity of the data structure, and their exponential complexity is practically prohibitive.

Table 3. General data and initial conditions.

<table>
<thead>
<tr>
<th>Prediction horizon $N_p$</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control horizon $N_c$</td>
<td>3</td>
</tr>
<tr>
<td>Sample time $T$</td>
<td>1 s</td>
</tr>
<tr>
<td>Simulation time</td>
<td>75 s</td>
</tr>
<tr>
<td>Number of variables</td>
<td>MLD: 24, Linear: 12</td>
</tr>
<tr>
<td></td>
<td>of which dummy: 9</td>
</tr>
<tr>
<td></td>
<td>of which integer: MLD: 3, Linear: --</td>
</tr>
<tr>
<td>Number of parameters</td>
<td>11</td>
</tr>
<tr>
<td>Number of constraints (MLD)</td>
<td>85 (89 for method 5)</td>
</tr>
<tr>
<td>Number of constraints (Linear)</td>
<td>78</td>
</tr>
<tr>
<td>Input initial condition</td>
<td>$u(-1) = 0$</td>
</tr>
<tr>
<td>State initial condition</td>
<td>$x(0) = [0, 5]_T$</td>
</tr>
<tr>
<td>State past condition</td>
<td>$x(-1) = [-5, 5, 3]_T$</td>
</tr>
</tbody>
</table>
for a high number of variables. In a further development we intend to consider also a model with discrete input, the gear shift. We will also consider the performance of some other interesting techniques, (Hedlund and Rantzer, 1999; Rakovic et al., 2004), that appear to be suitable for the described problem.

**Acknowledgment.** Authors are thankful to M. Kvasnica (ETH Zürich) for his important advices with the minimization tools and to M. Lazar (TU Eindhoven) for providing the end point constraint for Method 5.

### REFERENCES


### Table 4. Methods applied on the ACC design, with a constant velocity reference.

<table>
<thead>
<tr>
<th>Method</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time on-line (avg) (s)</td>
<td>0.09</td>
<td>0.90</td>
<td>0.19</td>
<td>1.16</td>
<td>0.09</td>
</tr>
<tr>
<td>Time on-line (max) (s)</td>
<td>0.57</td>
<td>1.64</td>
<td>0.16</td>
<td>8.31</td>
<td>0.18</td>
</tr>
<tr>
<td>Time off-line (s)</td>
<td>0.16</td>
<td>$6.84 \times 10^3$</td>
<td>0.002</td>
<td>$1.13 \times 10^5$</td>
<td>0.21</td>
</tr>
<tr>
<td>Memory usage on-line (Mb)</td>
<td>4.43+optimizer</td>
<td>28.5</td>
<td>3.5+optimizer</td>
<td>62</td>
<td>4.53+optimizer</td>
</tr>
<tr>
<td>Memory usage off-line (Mb)</td>
<td>0.05</td>
<td>28.5</td>
<td>0.01</td>
<td>62</td>
<td>0.06</td>
</tr>
<tr>
<td>Infeasibility (%)</td>
<td>62</td>
<td>1.3</td>
<td>60</td>
<td>27</td>
<td>100</td>
</tr>
<tr>
<td>Cost of evolution</td>
<td>337.71</td>
<td>$5.38 \times 10^3$</td>
<td>474.15</td>
<td>365.32</td>
<td>–</td>
</tr>
<tr>
<td>Maximum tractable $N_p$</td>
<td>18</td>
<td>2</td>
<td>&gt;30</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>Number of regions</td>
<td>–</td>
<td>4541</td>
<td>–</td>
<td>19311</td>
<td>–</td>
</tr>
</tbody>
</table>

### Table 5. Methods applied on the ACC design, with a time-varying velocity reference.

<table>
<thead>
<tr>
<th>Method</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time on-line (avg) (s)</td>
<td>0.09</td>
<td>1.13</td>
<td>0.19</td>
<td>1.79</td>
<td>0.07</td>
</tr>
<tr>
<td>Time on-line (max) (s)</td>
<td>0.17</td>
<td>1.91</td>
<td>0.39</td>
<td>8.56</td>
<td>0.44</td>
</tr>
<tr>
<td>Time off-line (s)</td>
<td>0.09</td>
<td>$6.84 \times 10^3$</td>
<td>0.003</td>
<td>$1.13 \times 10^5$</td>
<td>0.21</td>
</tr>
<tr>
<td>Memory usage on-line (Mb)</td>
<td>4.45+optimizer</td>
<td>28.5</td>
<td>3.5+optimizer</td>
<td>62</td>
<td>4.45+optimizer</td>
</tr>
<tr>
<td>Memory usage off-line (Mb)</td>
<td>0.05</td>
<td>28.5</td>
<td>0.01</td>
<td>62</td>
<td>0.05</td>
</tr>
<tr>
<td>Infeasibility (%)</td>
<td>84</td>
<td>1.3</td>
<td>77</td>
<td>56</td>
<td>100</td>
</tr>
<tr>
<td>Cost of evolution</td>
<td>334.68</td>
<td>$1.59 \times 10^3$</td>
<td>456.83</td>
<td>391.41</td>
<td>–</td>
</tr>
<tr>
<td>Maximum tractable $N_p$</td>
<td>15</td>
<td>2</td>
<td>&gt;30</td>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>Number of regions</td>
<td>–</td>
<td>4541</td>
<td>–</td>
<td>19311</td>
<td>–</td>
</tr>
</tbody>
</table>