Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

An MPC design flow for automotive control and applications to idle speed regulation

S. Di Cairano, D. Yanakiev, A. Bemporad, I.V. Kolmanovsky, D. Hrovat

Abstract— **This paper describes the steps of a model predictive control (MPC) design procedure developed for a broad class of control problems in automotive engineering. The design flow starts by deriving a linearized discrete-time prediction model from an existing simulation model, augmenting it with integral action or output disturbance models to ensure offset-free steady-state properties, and tuning the resulting MPC controller in simulation. Explicit MPC tools are employed to synthesize the controller to quickly assess controller complexity, local stability of the closed-loop dynamics, and for rapid prototype testing. Then, the controller is fine-tuned by refining the linear prediction model through identification from experimental data, and by adjusting from observed experimental performance the values of weights and noise covariances for filter design. The idle speed control (ISC) problem is used in this paper to exemplify the design flow and our vehicle implementation results are reported.**

I. Introduction

Model predictive control (MPC) [1], [2] is an optimization-based closed-loop control strategy for shaping the response of a process through minimization of a performance criterion and for handling constraints on system inputs, states, and outputs. The optimal input sequence is computed during each control update cycle by optimizing the predicted system trajectory, computed through a suitable *prediction model*. The MPC controller can be synthesized as a piecewise linear state-feedback control [3], with no need for on-line optimization, and processes with fast dynamics, such as the ones in automotive applications, can be addressed, see [4]–[6] for some examples.

We describe an MPC design flow that is composed of a sequence of procedural steps which exploit both simulation models and experimental data in order to tune the behavior of the MPC controller in a way that leads to fast prototyping and experimental evaluation. We suggest to first use a simulation model of the process, which is normally available if a model-based control system design approach is followed, and to generate from it a suitable prediction model for the MPC controller. The MPC controller based on such a model can be evaluated through simulations. For the actual implementation, an explicit model predictive controller [3] can be synthesized. The controller complexity can be easily assessed, and, if needed, the prediction model structure can be revised to trade-off controller performance versus complexity. Furthermore, the stability of the closed-loop dynamics and the region where feasible commands are issued can be easily analyzed. After a satisfactory complexity-performance balance has been found, the prediction model has to be tuned by means of experimental data. The effect of noise and disturbances has to be evaluated and quantified to be used in estimator design. Finally, the source code of a candidate set of controllers is automatically generated and experimentally validated.

In this paper, the design flow is illustrated on the idle speed control (ISC) problem. For this problem we design and synthesize MPC controllers of different structure, complexity and performance, and we discuss their implementation on the vehicle.

Historically, idle speed control is related to one of the oldest closed-loop systems discussed in the controls literature, the so called Watt's governor (1787), which may be viewed as a speed controller for a steam engine. The continuing interest in ISC fo spark-ignition engines is motivated by the opportunities to reduce fuel consumption at idle by lowering idle speed set-point and reducing spark reserve. A higher performing ISC based on MPC techniques may be able to avoid engine stalls even with lower idle speed set-point and spark reserve, thereby enabling substantial fuel consumption improvements.

To the best of the authors' knowledge, an MPC design for the ISC problem was first proposed in [7]. At the time this approach could not have been easily implemented, as it required an optimization problem to be solved on-line in real-time. In this paper, we complete the development and implementation of an MPC solution for ISC by exploiting the newly available techniques to compute the MPC law as an explicit state feedback. In this form, the MPC solution is easily implemented in the electronic control unit (ECU).

A. Notation

$\mathbb{R}$ and $\mathbb{Z}$ denote the set of real and integer numbers, respectively. $\mathbb{Z}_{0+}$ denotes the set of nonnegative integer numbers and $\mathbb{Z}_{[a,b]}$ denotes the set $\{r \in \mathbb{Z} : a \leq r \leq b\}$. I_n is the identity matrix of dimension n, and $\mathbf{1}$ and $\mathbf{0}$ are vectors or matrices entirely composed of 1 and 0, respectively. Given a vector a, $[a]_i$ is the i^{th} coordinate. Relational operators are applied componentwise to vectors.

II. Model-based Design: Prediction Model

Model-based design, in which the control system is developed with the help of a dynamical model of the process, is

A. Bemporad and S. Di Cairano are with Dipartimento di Ingegneria dell'Informazione, Università di Siena, bemporad,dicairano@dii.unisi.it

D. Yanakiev, I.V. Kolmanovsky and D. Hrovat are with Ford Motor Company, Dearborn, Michigan, USA. dyanakie,ikolmano,dhrovat@ford.com

S. Di Cairano has recently joined Ford Motor Company, sdicaira@ford.com

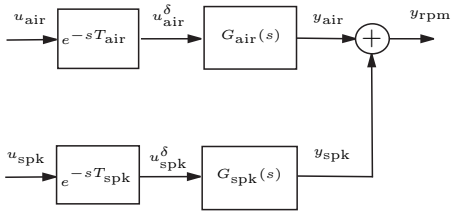


Fig. 1. Prediction model of the engine for idle speed control design

an evolving trend in automotive engineering. The dynamical model is usually derived from first principles, and further refined for consistency with experimental data and strategies implemented in the ECU. For the ISC problem considered in this paper, a nonlinear engine and crankshaft model that represents engine speed dynamics and torque converter behavior was used as a starting point.

In the ISC problem, the manipulated inputs are the spark advance $u_{\rm spk}$ and the airflow $u_{\rm air}$, and the controlled output is the engine speed $y_{\rm rpm}$. The spark advance command is measured in degrees [deg], and it is commanded with respect to a nominal set-point for idle, which allows about 15 degrees reserve with respect to maximum break torque (MBT) spark. For our problem, $u_{\rm spk}$ is thus constrained in the interval $[-15, 15]$deg. The airflow command is measured in pounds per minute and it is normalized to the interval $[0, 10]$. The engine speed is measured in revolutions per minute [rpm].

A set of specifications is given. The idle engine speed reference is 650 rpm, and the engine speed must remain in the interval $[450, 2000]$ rpm. The idle speed controller runs with a sampling period $T_s = 30$ ms. Only the signals $u_{\rm air}$, $u_{\rm spk}$, and $y_{\rm rpm}$ are available for closing the control loop.

Frequently, the simulation model is very detailed, and it must be approximated to obtain a prediction model that is simple enough to yield a tractable MPC optimization problem, but that still captures the most significant dynamics. As discussed in [8], for idle speed control the engine model can be reduced to two transfer functions with delays, one from each input to the output, see Figure 1. The transfer function parameters can be easily obtained by running the simulation model. The delays are approximately $T_{\rm air} = 0.12$s, $T_{\rm spk} = 0.03$s at 650 rpm.

A. Transfer function from delayed inputs

We first estimate the transfer function from the delayed input to the output, that is, from $u^{\delta}_{\rm air}$ to $y_{\rm air}$, and from $u^{\delta}_{\rm spk}$ to $y_{\rm spk}$ in Figure 1. As pointed out in [8], the transfer function from the delayed airflow to the engine speed has no zeros, while the transfer function from the delayed spark advance to the engine speed has a stable zero. Hence,

$$G_{\rm air}(s) = k_1 \frac{1}{\frac{s^2}{\omega_1} + 2\frac{\delta_1}{\omega_1}s + \omega_1^2}, \tag{1a}$$

$$G_{\rm spk}(s) = k_2 \frac{\frac{s}{a} + 1}{\frac{s^2}{\omega_2} + 2\frac{\delta_2}{\omega_2}s + \omega_2^2}. \tag{1b}$$

The parameters k_i, δ_i, ω_i, $i = 1, 2$, and a can be estimated by analyzing the step and frequency responses of the engine model running at 650 rpm, which is our linearization point for linearization input $u_{\rm spk} = 0$, $u_{\rm air} = 2.13$. After sampling with period $T_s = 30$ ms and converting to discrete-time state-space form, the estimated transfer functions are expressed as

$$\begin{aligned} x_{\rm air}(k+1) &= A_{\rm air}x_{\rm air}(k) + B_{\rm air}u^{\delta}_{\rm air}(k), & \text{(2a)}\\ y_{\rm air}(k) &= C_{\rm air}x_{\rm air}(k), & \text{(2b)} \end{aligned}$$

$$\begin{aligned} x_{\rm spk}(k+1) &= A_{\rm spk}x_{\rm spk}(k) + B_{\rm spk}u^{\delta}_{\rm spk}(k) & \text{(3a)}\\ y_{\rm spk}(k) &= C_{\rm spk}x_{\rm spk}(k), & \text{(3b)} \end{aligned}$$

where $x_{\rm air}, x_{\rm spk} \in \mathbb{R}^2$.

B. Delay modeling

The effect of the delay can be accounted for in the prediction model in several ways. The approach of this paper is to model in discrete time the transfer function from the delayed inputs $u^{\delta}_{\rm air}(x)$, $u^{\delta}_{\rm spk}(x)$, and then to introduce the delays as additional states.

In the ISC problem, $n^{\delta}_{\rm air} = \frac{T_{\rm air}}{T_s} = 4$ and $n^{\delta}_{\rm spk} = \frac{T_{\rm spk}}{T_s} = 1$ are integer valued. The discrete-time model of a generic signal $u(\cdot)$ delayed by $n^{\delta} \in \mathbb{Z}_{0+}$ steps is $x^{\delta}(k+1) = A^{\delta}x^{\delta}(k) + B^{\delta}u(k)$, $u^{\delta}(k) = C^{\delta}x^{\delta}(k)$, where $x^{\delta} \in \mathbb{R}^{n^{\delta}}$ and

$$A^{\delta} = \begin{bmatrix} 0 & \cdots & 0 \\ & & \vdots \\ & I_{n_{\delta}-1} & 0 \end{bmatrix}, \quad \begin{aligned} B^{\delta} &= [1\ 0 \cdots 0]^T, \\ C^{\delta} &= [0 \cdots 0\ 1]. \end{aligned}$$

Let

$$\begin{aligned} x^f_{\rm air}(k+1) &= A^f_{\rm air}x^f_{\rm air}(k) + B^f_{\rm air}u_{\rm air}(k) & \text{(4a)}\\ y^f_{\rm air}(k) &= C^f_{\rm air}x^f_{\rm air}(k), & \text{(4b)} \end{aligned}$$

$$\begin{aligned} x^f_{\rm spk}(k+1) &= A^f_{\rm spk}x^f_{\rm spk}(k) + B^f_{\rm spk}u_{\rm spk}(k) & \text{(5a)}\\ y^f_{\rm spk}(k) &= C^f_{\rm spk}x^f_{\rm spk}(k), & \text{(5b)} \end{aligned}$$

where $x^f_{\rm air} \in \mathbb{R}^6$, $x^f_{\rm spk} \in \mathbb{R}^3$, be the state-space models obtained, respectively, by cascading the fourth order state-space model of the airflow delay with (2), and by cascading the first order model of the spark delay with (3). The resulting linear model of the engine is

$$x^p(k+1) = A^p x^p(k) + B^p u^p(k), \tag{6a}$$

$$y_{\rm rpm}(k) = C^p x^p(k), \tag{6b}$$

$$x^p = \begin{bmatrix} x^f_{\rm air} \\ x^f_{\rm spk} \end{bmatrix},\ u^p = \begin{bmatrix} u_{\rm air} \\ u_{\rm spk} \end{bmatrix},\ A^p = \begin{bmatrix} A^f_{\rm air} & 0 \\ 0 & A^f_{\rm spk} \end{bmatrix}, \tag{6c}$$

$$B^p = \begin{bmatrix} B^f_{\rm air} \\ B^f_{\rm spk} \end{bmatrix},\ C^p = \begin{bmatrix} C^f_{\rm air} & C^f_{\rm spk} \end{bmatrix}. \tag{6d}$$

where $x^p \in \mathbb{R}^9$, and $u^p \in \mathbb{R}^2$.

C. Models for disturbance rejection

The prediction model (6) is now extended with additional components that relate to the specifications. In particular, in the ISC problem offset-free tracking (rejection of constant disturbances) and zero spark deviation $u_{\rm spk} = 0$ are desired in steady state. In order to achieve disturbance rejection, one can introduce *integral action* [9] by adding the dynamics $q_{\rm rpm}(k+1) = q_{\rm rpm}(k) + T_s y_{\rm rpm}(k)$, where $q_{\rm rpm} \in \mathbb{R}$ is the discrete-time integral of the output (such an approach can be also applied for tracking time-varying reference signals, by integrating the tracking error). We use this approach also to guarantee that the value of $u_{\rm spk}$ is zero in steady-state. The resulting prediction model is

$$x(k+1) = Ax(k) + Bu(k), \tag{7a}$$
$$y(k) = Cx(k), \tag{7b}$$
$$x = \begin{bmatrix} x^p \\ q_{\rm rpm} \\ q_{\rm spk} \end{bmatrix},\ u = u^p,\ A = \begin{bmatrix} A^p & 0 & 0 \\ T_s C^p & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7c}$$
$$B = \begin{bmatrix} B^p \\ 0 \\ T_s \end{bmatrix},\ C = \begin{bmatrix} C^p & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7d}$$

An alternative approach to ensure offset-free tracking is to introduce a disturbance model [10]. In particular, in [10] the authors show that under mild technical conditions, output additive integrated white noise can be used as a disturbance model. An MPC controller whose prediction model is augmented with such a disturbance model guarantees asymptotic rejection of constant disturbances. The disturbance is estimated concurrently with the state vector, for instance by a Kalman filter based on the estimation model

$$x(k+1) = Ax(k) + Bu(k) + Ew(k), \tag{8a}$$
$$y(k) = Cx(k) + Dv(k), \tag{8b}$$
$$x = \begin{bmatrix} x^p \\ x^d \\ q_{\rm spk} \end{bmatrix},\ u = u^p,\ A = \begin{bmatrix} A^p & 0 & 0 \\ 0 & A^d & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{8c}$$
$$B = \begin{bmatrix} B^p \\ 0 \\ T_s \end{bmatrix},\ C = \begin{bmatrix} C^p & C^d & 0 \end{bmatrix}. \tag{8d}$$

where $x^d \in \mathbb{R}^{n_d}$ is the state of the disturbance model, A^d and C^d are the matrices of the disturbance model used for prediction, and in the case of integrated white noise $n_d = 1$, $A^d = 1$, and $C^d = 1$. In equation (8), $w(k)$ and $v(k)$ are Gaussian white noise vectors with zero mean and covariance matrices W and V, respectively. The noise vectors are assumed to be zero in prediction because of the certainty equivalence principle. However, they are important tuning knobs for the estimator, as explained later in Section IV, and, when the output disturbance model is used, they strongly affect the tracking and disturbance rejection performances. Instead, when integral action is used, the estimator noise covariances have smaller impact in disturbance rejection performance.

III. MPC DESIGN AND SYNTHESIS

Once a simple yet representative prediction model is available, a model predictive controller can be designed. The first step is to setup the MPC optimization problem

$$\min_{\sigma, \mathbf{u}(k)} \quad \omega\sigma^2 + \sum_{i=0}^{N-1} \|y(i|k) - r_y(k)\|_2^Q + \|\Delta u(i|k)\|_2^R \tag{9a}$$
$$\text{s.t.} \quad x(i+1|k) = Ax(i|k) + Bu(i|k),\ i \in \mathbb{Z}_{[0,N-1]} \tag{9b}$$
$$y(i|k) = Cx(i|k),\ i \in \mathbb{Z}_{[0,N-1]} \tag{9c}$$
$$u_{\min} \le u(i|k) \le u_{\max},\ i \in \mathbb{Z}_{[0,N-1]} \tag{9d}$$
$$\Delta u_{\min} \le \Delta u(i|k) \le \Delta u_{\max},\ i \in \mathbb{Z}_{[0,N-1]} \tag{9e}$$
$$y_{\min} - \sigma\mathbf{1} \le y(i|k) \le y_{\max} + \sigma\mathbf{1}, i \in \mathbb{Z}_{[0,N_c-1]} \tag{9f}$$
$$u(i|k) = u(N_u - 1|k),\ i \in \mathbb{Z}_{[N_u-1,N-1]} \tag{9g}$$
$$u(-1|k) = u(k-1) \tag{9h}$$
$$x(0|k) = \hat{x}(k) \tag{9i}$$
$$\sigma \ge 0, \tag{9j}$$

where $\|a\|_2^Q = a^T Q a$, $\Delta u(i|k) = u(i|k) - u(i-1|k)$, $\hat{x}$ is the state estimate, r_y is the output reference, and $\mathbf{u}(k) = \{u(0|k), \ldots, u(N|k)\}$. $N \in \mathbb{Z}_+$ is the prediction horizon, $N_c \le N$ the constraint horizon, and $N_u \le N$ is the control horizon. In problem (9), (9a) represents the performance criterion to be optimized, and (9b), (9c) are the prediction model for idle speed control, namely either (7) or (8) (where $w(k) = \mathbf{0}$, $v(k) = \mathbf{0}$). Constraints (9d), (9e) and (9f) are bounds on input, input variation, and output. In particular, (9f) defines *soft output constraints*, that prevent problem (9) to be infeasible, so that the control action is always defined. Soft constraints can be violated at the price of a penalty, modelled by the optimization variable $\sigma \in \mathbb{R}$, weighted by the large (i.e., at least two orders of magnitude larger than the other weights) positive constant ω. Finally, (9g) limits the free moves to N_u, and (9h) and (9i) set the previously applied input value, and the initial state to the value of the state estimate at time k, respectively.

Algorithm 1 (implicit MPC): At each control cycle k do:

1. read the measurements and compute the state estimate $\hat{x}(k)$;
2. solve the quadratic programming problem (9) to obtain the optimal input sequence $\mathbf{u}^*(k)$;
3. apply the input $u(k) = u^*(0|k)$;

The MPC controller that implements Algorithm 1 is tuned in closed-loop with the simulation model. This process requires adjusting the weights ω, Q, R and the horizons N, N_u, N_c in problem (9) and, possibly, revising the prediction model, if a satisfactory performance is not achieved.

A. Explicit MPC synthesis and controller analysis

To avoid ECUs in commercial vehicles to run optimization software, the explicit implementation of the MPC controller [3] has to be computed. The explicit MPC is an alternative way of evaluating the control action on line. Multiparametric quadratic optimization is used off line to pre-compute the solution of problem (9) as a function of the parameters $\hat{x}(k)$, $u(k-1)$ and $r_y(k)$. This approach provides a piecewise affine continuous control law in the form

$$u(k) = F_i\hat{x}(k) + G_i u(k-1) + T_i r_y(k) \tag{10a}$$
$$i \in \mathbb{Z}_{[1,n_r]} : H_i\hat{x}(k) + J_i u(k-1) + K_i r_y(k) \le 0, \tag{10b}$$

where F_i, G_i, T_i, H_i, J_i, K_i are constant matrices of suitable dimensions for all $i \in \mathbb{Z}_{[1,n_r]}$. Inequalities (10b) partition the $(\hat{x}, u, r_y)$-space into n_r polyhedral regions, that is, for any value of $(\hat{x}, u, r_y)$ there is at most a value of $i \in \mathbb{Z}_{[1,n_r]}$ such that (10b) is satisfied, and (10a) defines the affine state feedback control law associated to each region $i \in \mathbb{Z}_{[1,n_r]}$. Control law (10) describes the region where (9) has a feasible solution. Since soft constraints are used, for all values $(\hat{x}(k), u(k-1), r_y(k))$ there exists $\bar{\imath} \in \mathbb{Z}_{[1,n_r]}$ such that (10b) is satisfied for $i = \bar{\imath}$ by $(\hat{x}(k), u(k-1), r_y(k))$.

Algorithm 2 (explicit MPC): At each control cycle k, perform the following:

1. read the measurements and compute the state estimate $\hat{x}(k)$;
2. search for $\bar{\imath} \in \mathbb{Z}_{[1,n_r]}$ such that (10b) is satisfied;
3. compute $u(k)$ by evaluating (10a) for $i = \bar{\imath}$;

Algorithm 2 generates the same control input as Algorithm 1, but has the following advantages: (i) it does not need an optimization code to be executed in the ECU, (ii) the time required to compute the command is in general much smaller, especially for small MPC problems [2], and (iii) the worst-case computation time of the controller is predictable. The operations to compute the command from the explicit MPC law (10) are sums, products, and comparisons in known quantities. Thus, the worst case number of operations to be executed per control cycle can be computed. Such number of operations is often very conservative (see [6]), but it can be still used as a conservative bound in the control architecture design. Similarly, the memory usage of the controller can be quantified, since the executed code is simple and fixed, and the stored data are the matrices F_i, G_i, T_i, H_i, J_i, K_i, for $i \in \mathbb{Z}_{[1,n_r]}$.

For idle speed control we have set $N = 30$, $N_u = 2$, $N_c = 2$ in (10), where we have used (7) as the prediction model. The constraints are defined by (9d)–(9f), where

$$u_{\max} = \begin{bmatrix} 10-u_{\rm FF} \\ 15 \end{bmatrix},\ u_{\min} = \begin{bmatrix} 0-u_{\rm FF} \\ -15 \end{bmatrix},$$
$$y_{\max} = \begin{bmatrix} 2\cdot 10^3 - y_{\rm ss} \\ \infty \\ \infty \end{bmatrix},\ y_{\min} = \begin{bmatrix} 450-y_{\rm ss} \\ -\infty \\ -\infty \end{bmatrix},$$
$$\Delta u_{\max} = -\Delta u_{\min} = \begin{bmatrix} \infty \end{bmatrix},$$

where $y_{\rm ss} = 650$ rpm is the engine speed around which the linear prediction model is estimated, and $u_{\rm FF}$ is the corresponding airflow equilibrium input for $u_{\rm spk} = 0$, that is sent to the engine as a feedforward input. For $y_{\rm ss} = 650$ rpm, the correct value of the feedforward is $u_{\rm FF} = 2.13$. After performing some tests with the implicit MPC algorithm, we tuned $Q = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 5 \end{bmatrix}$, $R = I_2$, and computed the explicit control law (10) using the toolbox [11]. In this case, $n_r = 131$, and the partition is over a 16-dimensional space: 11 components are the estimated state vector (6 from (4), 3 from (5) and 2 from the integral action on $y_{\rm rpm}$ and $u_{\rm spk}$), 2 are previous input values, and 3 are references (engine speed, integral of engine speed tracking error, integral of spark command).

For the above controller the maximum number of operations per control cycle is less than $45 \cdot 10^3$, which leads to $1.5 \cdot 10^6$ operations per second. By considering fixed reference values, the operation number is reduced by about 20%. The data memory usage for the controller is about 100 KB. For conventional ECUs, the MPC controller use in the worst case $1.5\% - 5\%$ of the total computational power, resulting feasible. In fact, at idle the ECUs are under-utilized, due to low activation rate of tasks triggered by engine events, hence the needed computational power is available.

The MPC controller based on (9) does not guarantee closed-loop stability. Stability guarantees can be obtained by applying the techniques surveyed in [12]. However, those typically require long prediction horizons and hence generate complex controllers.

If the origin is in the interior of the feasible region of problem (9), there exists an index $\bar{\imath} \in \mathbb{Z}_{[1,n_r]}$ such that the origin is also in the interior of the polyhedral region described by (10b) for $i = \bar{\imath}$. Thus, in a neighborhood of the origin, the explicit controller (10) is equal to the linear state feedback (10a) where $i = \bar{\imath}$, and it can be proven that $G_{\bar{\imath}} = \mathbf{0}$. As a result, considering $r_y(k) = 0$ for all $k \in \mathbb{Z}_{0+}$, the local stability of the closed-loop dynamics can be studied by analyzing the system

$$x(k+1) = Ax(k) + Bu(k) \quad (12a)$$
$$u(k) = F_{\bar{\imath}}x(k) + G_{\bar{\imath}}u(k-1). \quad (12b)$$

By calling $\chi(x) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}$, (12) is asymptotically stable if the spectral radius ρ of the matrix $A_{\rm cl} = \begin{bmatrix} A+BF_{\bar{\imath}} & BG_{\bar{\imath}} \\ F_{\bar{\imath}} & G_{\bar{\imath}} \end{bmatrix}$ is smaller than 1. For the idle speed control problem, in the MPC tuning described before, $\rho(A_{\rm cl}) = 0.9726$, ensuring that the system is locally stable.

A global stability analysis can be performed a posteriori by considering the piecewise affine (PWA) [13] constructed from the prediction model (9b) in closed-loop with the explicit MPC (10), and using the tools described in [14].

B. Closed-loop simulations

Closed-loop simulation tests were performed in SIMULINK® with the MPC in closed-loop with the engine simulation model. The feedforward on the airflow is $u_{\rm FF} = 2.5$, a value slightly different from the one that achieves stabilization at 650 rpm, in order to test the regulation capabilities of the controller and the effects of a nonzero steady-state value of $u_{\rm air}$.

The controller is enabled at $t = 5$ when the initial transient of the engine is completed, and it first has to regulate the system to 650 rpm. Then, a disturbance rejection test is performed by introducing an unmeasured torque disturbance load τ_d [Nm] of 20 Nm that is subtracted from the engine generated torque, starting at $t = 8$ s, and ending at $t = 17$ s, simulating the effect of power steering.

Figure 2 shows the resulting closed-loop behavior. Note that both inputs reach saturation, and that the spark input remains saturated at its maximum until the disturbance has been almost completely rejected.

The behavior shown in Figure 2 can be changed by tuning the weights Q, R, in (9a). For instance, an overdamped

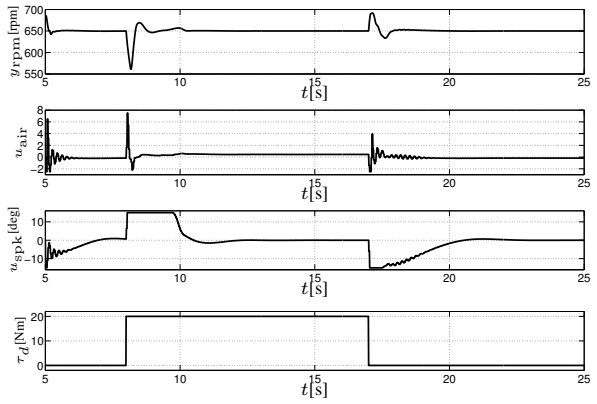


Fig. 2. Simulation of the MPC controller for idle speed in closed-loop with the SIMULINK engine model

behavior, where the engine speed settles more slowly without overshoot, can be obtained. The controller tuning is in general selected by considering requirements such as driver's feelings, effects on engine wear, and fuel consumption.

IV. DATA-BASED TUNING AND EXPERIMENTS

The model-based design described in Section III is important to evaluate whether the specifications can be met. In particular, it allows one to assess whether the prediction model is detailed enough, and also, through the explicit controller computation, the worst-case number of operations per control cycle, and the closed-loop (local) stability.

However, when implementing the controller on the real process, the design may need to be revised. First of all it is possible that the real process dynamics are quantitatively different from the simulated dynamics. Also, in a real process there are several disturbances acting, including sensor noise and disturbances on the actuators. Thus, a fundamental role is played by the estimation algorithm, that trades off between noise rejection and convergence speed of the estimator.

For the experimental implementation we chose to apply MPC only to the throttle input and retained the existing control loop (of Proportional plus Derivative type) for spark. With this approach, the implementation is further simplified while the loss in performance is not greatly appreciable. The MPC design strategy easily allows one to reconfigure the design to use only the airflow as controlled variable, by simply removing $u_{\rm spk}$ from problem (9), and the corresponding terms in the cost functions and constraints. The plant model (6) is now composed only of the delayed state space model from the airflow to the engine speed (4) (the integral action on the input $u_{\rm spk}$ has been obviously removed from prediction model (7)). The resulting MPC controller is significantly simpler ($n_r = 7$ regions in a 10-dimensional space), and the worst case number of operations per control cycle of the explicit controller is only 450.

In order to account for the modified plant, composed of the engine in closed loop with the spark controller, we used experimental data to tune the numerical parameters of prediction model (9b), (9c), without changing its complexity. After re-tuning the MPC with the new model (and repeating the stability analysis of Section III-A), the controller has been tested in simulation against disturbances. We have applied additive input disturbances and sensor disturbances to evaluate the closed-loop behavior of the controller and to tune the parameters of the state estimator. In simulation we have used the linearized prediction model refined by experimental data as plant model, since the focus of this test is the behavior with respect to noise and disturbances. This approach becomes particularly important if the disturbance model approach in [10] is used, since the trade-off between steady-state convergence speed and noise filtering is set by the estimator parameters. In the idle speed problem, we use the stationary Kalman filter

$$\hat{x}(k+1) = [I - K_f C](A\hat{x}(k) + B\hat{u}(k)) + K_f y(k+1) \quad (13)$$

where A, B, and C are the prediction model matrices, and $K_f = PC^T(CPC^T + V)^{-1}$ is the stationary filter gain constructed from the solution P of the Riccati equation

$$P = APA^T + W - APC^T(CPC^T + V)^{-1}CPA^T. \quad (14)$$

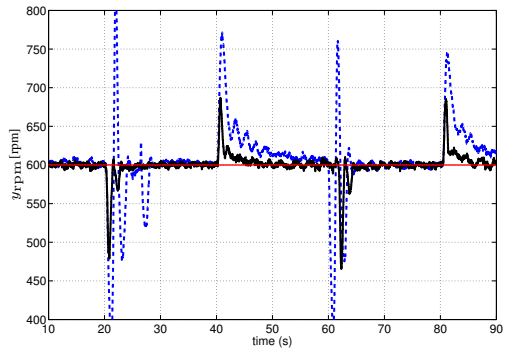
The tuning parameters of the Kalman filter are the matrices W and V, that in the case of a linear system subject to Gaussian noise are set equal to the process noise and to the sensor noise, respectively, for (13) to result in the optimal estimator. In practice, it is very difficult to quantify W, V a priori, as the plant dynamics are not purely linear and the noise is not exactly white and Gaussian. Hence, the matrices have to be tuned according to standard rules. When $\|W\| \gg \|V\|$, the observer relies on measurements, and it converges faster. When $\|W\| \ll \|V\|$, the measurements are not trusted, causing slow estimate convergence, but also reduced sensitivity to measurement noise.

A. Experimental results for idle speed MPC control

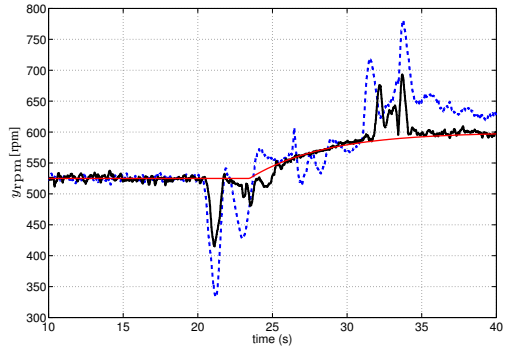
We have tested several MPC controllers in a vehicle, in order to find the best tradeoff between closed-loop convergence performance, robustness, and actuator excitation. We mainly performed disturbance rejection tests, since stall avoidance, and hence quick and robust disturbance rejection, is the most critical requirement for idle speed control.

The test vehicle is a Ford pickup truck with a V8 4.6L gasoline engine, equipped with a developmental ECU. The C-code for the MPC controllers is automatically generated by the HYBRID TOOLBOX [11], and compiled in a dSPACE® rapid prototyping system connected to the car's ECU. The MPC controller is compared with an available baseline controller implemented in the ECU. The reference idle speed r_y is controlled by the ECU and may vary depending on the engine conditions.

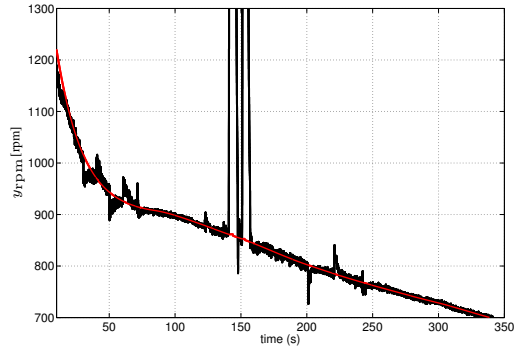
The results of three experiments are reported in Figure 3, where the MPC controller (solid line) is compared to the baseline controller (dashed line) in different disturbance rejection tests (solid thin line, the idle speed reference). In Figure 3(a), the vehicle is in neutral gear, and the torque disturbances are obtained by activating power steering (PS) at the end of the steering wheel travel. The power steering



(a) Disturbance rejection test (PS)



(b) Double disturbance rejection test (PS+AC)



(c) Disturbance rejection (PS,AC,tip-in) during cold start

Fig. 3. Experimental results of the MPC for idle speed control in the test vehicle, and comparison with a baseline controller.

is activated at $t = 20$s and $t = 60$s. The engine speed drop is reduced by 50% and the convergence to the setpoint is about 10s faster. Figure 3(b) shows the results of a double disturbance test, where PS and air conditioning (AC) are activated at $t = 20$s, and they are deactivated at $t = 31$s. This test is executed with vehicle in drive gear, where the idle sped reference is lower (525 rpm), and also the engine speed lower bound for the MPC is reduced (350 rpm). The activation/deactivation of AC is delayed by a couple of seconds by the vehicle ECU, so that in fact the disturbances occur sequentially (AC becomes active around $t = 22$s, and inactive around $t = 33$s). The AC activation causes the idle speed reference to increase to 600rpm. In this test the superior performance of MPC is even more evident. The MPC not only attenuates the engine speed drop due to the disturbance, it is also much faster in recovering the setpoint. When the second disturbance occurs the MPC has already recovered from the first one. This is not the case for the baseline controller, and as a consequence it oscillates heavily. Note also that by the end of the test the baseline controller has not reached yet the setpoint, while the MPC has settled more than 5s before. Finally, in Figure 3(c) a cold start test where the idle speed reference progressively decreases is shown. Several PS and AC disturbances are introduced, as well as two driver tip-in (around $t = 150$s). Even if the engine speed is far from the linearization value (650rpm) the MPC controller can handle the model variation.

V. Discussion

We have proposed a design flow to develop model predictive controllers targeted to automotive applications, that makes use of simulation model and experimental data to tune the different parameters of the controller. The procedure has been applied to idle speed control. An improved idle speed controller can reduce the need for a spark reserve, and hence consistently improve the fuel efficiency. The results show that the MPC largely outperforms the available baseline controller.

References

[1] J. Maciejowski, *Predictive control with constraints*. Englewood Cliffs, NJ: Prentice Hall., 2002.

[2] A. Bemporad, "Model-based predictive control design: New trends and tools," in *Proc. 45th IEEE Conf. on Decision and Control*, San Diego, CA, 2006, pp. 6678–6683.

[3] 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, 2002.

[4] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat, "An MPC/hybrid system approach to traction control," *IEEE Trans. Contr. Systems Technology*, vol. 14, no. 3, pp. 541–552, 2006.

[5] P. Ortner and L. del Re, "Predictive Control of a Diesel Engine Air Path," *IEEE Trans. Contr. Systems Technology*, vol. 15, no. 3, pp. 449–456, 2007.

[6] S. Di Cairano, A. Bemporad, I. Kolmanovsky, and D. Hrovat, "Model predictive control of magnetically actuated mass spring dampers for automotive applications," *Int. J. Control*, vol. 80, no. 11, pp. 1701–1716, 2007.

[7] D. Hrovat, "MPC-based idle speed control for IC engine'," in *Proc. of FISITA conference*, Prague, Czech Rep., 1996.

[8] D. Hrovat and J. Sun, "Models and control methodologies for IC engine idle speed control design," *Control Engineering Practice*, vol. 5, no. 8, pp. 1093–1100, 1997.

[9] H. Kwakernaak and R. Sivan, *Linear optimal control systems*. New York: Wiley-Interscience, 1972.

[10] G. Pannocchia and J. Rawlings, "Disturbance models for offset-free model-predictive control," *AIChE Journal*, vol. 49, no. 2, pp. 426–437, 2003.

[11] A. Bemporad, *Hybrid Toolbox – User's Guide*, Dec. 2003, http://www.dii.unisi.it/hybrid/toolbox.

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

[13] A. Bemporad, W. Heemels, and B. D. Schutter, "On hybrid systems and closed-loop MPC systems," *IEEE Trans. Automatic Control*, vol. 47, no. 5, pp. 863–869, 2002.

[14] G. Ferrari-Trecate, F. Cuzzola, D. Mignone, and M. Morari, "Analysis of discrete-time piecewise affine and hybrid systems," *Automatica*, vol. 38, pp. 2139–2146, 2002.